Automated support of regulated data exchange

Dijkstra, Pieter

Automated Support of Regulated Data Exchange

A Multi-Agent Systems Approach

RIJKSUNIVERSITEIT GRONINGEN

Automated Support of Regulated Data Exchange

A Multi-Agent Systems Approach

Proefschrift

ter verkrijging van het doctoraat in de
Rechtsgeleerdheid
aan de Rijksuniversiteit Groningen
op gezag van de
Rector Magnificus, dr. E. Sterken,
in het openbaar te verdedigen op
donderdag 19 april 2012
om 16.15 uur

door

Pieter Dijkstra
geboren op 9 oktober 1972
te Achtkarspelen

Promotor:	Prof. dr. H. Prakken
Copromotor:	Dr. mr. C.N.J. de Vey Mestdagh
Beoordelingscommissie:	Prof. T.J.M. Bench Capon Prof. dr. J.C. Hage Prof. dr. H.J. van den Herik

ISBN/EAN: 978-90-367-5471-2

This research is part of the ANITA (Administrative Normative Information Transaction Agents) research project[1]*, which aims at performing the fundamental research needed to develop a multi-agent system for regulated information exchange in the police intelligence domain (De Vey Mestdagh 2003)*

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project number 634.000.017

[1] For more information see http://www.nwo.nl/projecten.nsf/pages/1600112148/; accessed on July16th, 2011.

Contents

Chapter 1

Introduction

Many organizations (such as hospitals, tax authorities, and police departments) exchange data, an activity which is often regulated by law. In most cases there is a central institution (for example, the mother company or the ministry of Justice) that is interested in both optimal and legitimate data exchange, because it has to give account of the effectiveness and lawfulness of its operations to the outside world (for example, the parliament or the shareholders). Besides the central institution there are regionally or functionally distributed local institutions with their own interests. The central institutions take the interests at the distributed level into account by formulating legal norms and central policies, which give room for fine tuning in local policies and individual decisions. This is done by granting discretionary authority to local institutions.

The regulation of data exchange serves several objectives. On the one hand, the privacy of the persons who are the subjects of the data must be protected. On the other hand, the legitimate goals of the exchanging organizations must be served. Typically, organizations must balance the goal to execute their tasks by exchanging data with the goal to protect their interests at the local level by not exchanging confidential data, while staying within the law. For example, in the Dutch police domain, departments exchange data in order to execute the department's task to solve crime cases. However, when the exchanged data originates from informants, it is more difficult to obtain data from them when they know that they are not well protected. Moreover, while the police departments exchange data, departments have to comply with the applicable laws and regulations.

The balancing act between goals is caused by two characteristics of the applicable laws and regulations. First, the regulations give discretionary power to organizations in which types of situations organizations can decide (not) to exchange data. In those cases where it is permitted to exchange data, an organization member makes a decision which is expected to further his goals.

Second, the regulations use open textured concepts, which can have different local interpretations (for example 'the proper execution of the police task'). A concept is open textured if it is impossible to determine beforehand which situations can be classified as an instance of the concept.

In practice[2], data exchanging organizations can engage in dialogues and use their own interpretations of the regulations to further their goals. Therefore, during the interactions a definitive balance is sought between the goals of the organizations, which may give rise to interesting dialogues. For example, when an organization requests another organization to exchange data, the responding organization has to determine if its interests are served. The responding organization can negotiate by stating conditions under which it is willing to exchange data (for example that the data cannot be exchanged with other members). Furthermore, when a responding organization refuses a request, the requesting organization can try to persuade the other organization to exchange data. Ideally, these dialogues guarantee that an optimal and legitimate balance is found in the exchange of data characterized by different goals, which in some cases even conflict. However, in practice this ideal is not always realized: the norms from regulations are not well known and local policies are more aimed at the protection of local data, which has as consequences that only a part of what can be exchanged legitimately is actually exchanged (exchanges which are permitted do not occur in practice) and that sometimes data is exchanged illegitimately (exchanges which are forbidden do occur in practice).

Making sure that organizations stay within the law falls within regulatory compliance, which essentially means that organizations are aware of and take steps to comply with relevant laws and regulations. The need to improve the compliance with regulations was increased with a succession of corporate scandals such as the Enron disaster of 2001. Well-known regulations to strengthen regulatory compliance are (1) the Sarbanes-Oxley Act (SOX), which is a United States federal law to set standards for all U.S. public companies, (2) Basel II, which is an international standard for banking regulators, and (3) the EU Data Protection Directive, which regulates the processing of personal data within the European Union. One possible solution to support compliance with regulations is with legal knowledge-based systems. In practice, legal knowledge-based systems are mostly used to support application of regulations in public administration such as environmental permit law (De Vey Mestdagh 1998), income taxes and social

[2] Between 2002 and 2005 several interviews were conducted. For example, interviews with Tom van der Velde the privacy officer of the Dutch police officer in Groningen, conducted together with Wouter Teepe n 2002-2003; an interview with Doetse Huizinga together with Hugo Kielman and Wouter Koelewijn on 30-3-2005 in Apeldoorn; and several interviews with Paul Elzinga IT specialist of NPOL in 2005.

security (Van Engers et al. 2001). Furthermore, legal knowledge-based systems are also applied to support regulatory compliance (Hagerty 2007).
It is interesting to investigate to what extent the balancing act in regulated data exchange can be supported with advanced information technology. Knowledge-based systems can be used to support the application of regulations, such as in the above-mentioned public administration domain. However, knowledge-based systems are difficult to apply in domains where the regulations have many open-textured concepts and grant discretionary authorities Stranieri et al. (1999).

The distributed nature of many organizations suggests that it may be worthwhile to use multi-agent technology. A multi-agent system is a collection of artificial agents, where each agent has some degree of control over his actions and interacts with other agents to fulfill his own and/or the system's goals (Wooldridge 2002). A multi-agent system seems to provide a natural means to model organizations, where the agents represent the organization's members. Furthermore, organizations typically interact with each other to try to achieve their goals, which is also typical for agents in multi-agent systems.

The Dutch police domain provides a worthwhile example of the problem of regulated data exchange, as it illustrates that not always an optimal and legitimate balance is found in the exchange of data. The Dutch criminal investigation units balance the goal to execute their appointed police task by exchanging data with the goal to protect their investigations and informants by not disclosing data, while staying within the law (Koelewijn 2009, page 17). Since goals can conflict, officers of criminal investigation units confer whether severe-crime intelligence data can be exchanged (Cozijn 1996; Koelewijn 2009, page 128). However, in the Dutch police practice, the norms are not well known (Cozijn 1996; Schreuders and Wel 2005) and criminal investigation units prefer their goal to protect their own data (Kielman 2010) by refusing to exchange data. Therefore, in practice not enough legitimate and too much illegitimate data is exchanged between Dutch criminal investigation units. This thesis investigates the idea of utilizing multi-agent technology to support regulated data exchange between Dutch police departments.

1.1 Research context

To support automated regulated data exchange in the Dutch police domain, the applicable laws and regulations need to be analyzed and formalized, which entails (1) the representation of legal knowledge and (2) the application of that knowledge to concrete cases. Knowledge representation and reasoning with legal knowledge are two related subjects, which fall within the field of AI & Law research. The open-texturedness of legal concepts is a classic AI & Law issue (e.g., Bench-Capon

and Sergot 1989; Gardner 1987; McCarty 1980). Legal concepts are inherently open textured, because it is difficult to determine which instances fall under a legal concept.

Both the open-texturedness of legal concepts and the balancing act between different - possibly conflicting - goals can cause disagreements. Disagreements can occur during the internal reasoning and during the interactions between parties, which can both be modeled with argumentation. Argumentation (Bench-Capon and Dunne 2007; Rahwan and Simari 2009) is an important research area for this thesis because it captures two aspects in the legal domain: defeasible reasoning and the adversarial nature of legal argumentation. Defeasible reasoning is a form of reasoning in which conclusions can be withdrawn when new information becomes available. For example, a police officer reasons whether he can exchange intelligence data about the location of a drug house. Normally he reasons that the data can be exchanged. However, when the data is no longer up to date, the police officer also concludes that he cannot exchange the same data. Assuming that the reason for not exchanging data is stronger, he withdraws his previous conclusion. There are several approaches to formalize defeasible reasoning (McCarthy 1980; Reiter 1980). The argument-based approach is quite suitable to model legal reasoning because it uses concepts which are familiar in the legal domain, such as argument, counterargument, defeat, and dialogue (see Prakken and Vreeswijk 2002, and various chapters in Rahwan and Simari 2009 for overviews). In the argument-based approach, defeasibility is modeled as the interaction between conflicting arguments. For example, arguments can become defeated when they are attacked by other arguments. The argumentation-approach can also be used for argumentation between two parties (Amgoud et al. 2000; McBurney and Parsons 2002; Parsons et al. 2003). With dialogue game systems, argumentation between two parties is modeled as a game, where every utterance is seen as a move in the game. A dialogue game system regulates a dialogue by determining (1) the allowed moves which a party can make, (2) the termination, and (3) the outcome of a dialogue.

As discussed earlier, a multi-agent system is a natural way to model organizations which need to interact. In the field of multi-agent system research, argumentation has been gaining importance[3]. On the one hand, argumentation can be used for modeling, analyzing and implementing the internal reasoning of an agent. On the other hand, it can also be used to for facilitating the interactions between agents. Existing multi-agent system research, which applies argumentation on regulated data exchange does exist. For example, Doutre et al. (2005) and Perrussel et al.

[3] For instance, since 2004 there are the argumentation in multi-agent systems (ArgMas) workshops at the international conferences on autonomous agents and multi-agent systems (AAMAS).

(2007) view interactions about data exchange as argumentation and both use the medical domain as problem domain. However, Doutre et al.'s and Perrussel's research does not provide enough details to be directly usable for the present research (as will be explained in more detail in Chapter 7).
As for AI & Law research on regulatory compliance, this research is mainly focused on how to model and formalize regulations. For example, Giblin et al. (2005) propose a method in which a metamodel is used to formalize different regulations and to manage the regulatory requirements on businesses. A second example is Breaux and Vail (2006), who propose a method to extract rights and obligations from regulations and present how semantic models can be used to clarify ambiguities from the regulations. However, the AI & Law research on regulatory compliance concerns the application of knowledge-based systems. Recall that legal knowledge-based systems cannot be used in the problem domain of this thesis, which makes the research on regulatory compliance not suitable for this research.

There is a spectrum of different approaches to support automated regulated data exchange. At one end of the spectrum, there is Lessig's (1999) approach to embed legal concepts directly into the code (i.e., the source code of computer software), where software forces users to act according to the applicable regulations. A drawback of this approach only works in domains where the regulations have few open-textured terms and give few discretionary powers. At the other end, there is the more theoretical research from the AI & Law field, such as research on creating formal representations of values and on legal argumentation about finding balances between conflicting values or goals (Bench-Capon 2003; Bench-Capon and Sartor 2003; Sartor 2010). However, while this research deals with an interesting subject and is relevant in the problem domain, Bench-Capon's and Sartor's research is mainly theoretical and cannot be implemented directly. The approach by Buchanan et al. (2010) is a middle ground, because rather than formalizing the written regulations and the legal reasoning process, Buchanan et al. propose to mimic the outcomes of the legal reasoning process. In Buchanan et al.'s approach, exchanging organizations first have to agree which types of situations give access to which types of data. The agreements are the outcomes of the balancing act between different values or goals the organizations try to fulfill with exchanging data and are hard-coded into local rules, which are then automatically applied to information exchange requests. Buchanan et al.'s approach to support automated regulated data exchange is interesting because the hard-coded local rules can be used for an implementation. Moreover, their approach can be applied in domains where the regulations grant discretionary authorities and have many open-textured concepts. A drawback of this approach is that the balancing act is fixed for the agreed types of situations. Furthermore, with Buchanan et al.'s approach organizations do not reason with their interpretations of the applicable regulations about exchanging data

and cannot try to convince other organizations in case a request for data is rejected. Let us illustrate the possible drawbacks of Buchanan et al.'s approach with an example.

Assume that police departments agree that they do not exchange protected data, because they want to protect their informants. Furthermore, let us assume that data which reveals the location and owners of a large quantity of drugs is protected, since revealing the location will also expose the informant who has provided the data. If Buchanan et al.'s approach is used then the protected data will not be exchanged. However, in some situations protected data can be exchanged. For example, when the police department which requests the protected data knows that the drug location is leaked by a newspaper, the requesting department can use this knowledge to persuade the other department to exchange the data. If the responding police department is persuaded, the data could then be used to solve a crime case related to the drug location. In this example the police departments balance their goals during their interaction by not revealing the informant and executing their police task by helping to solve a crime case.

This thesis' approach to support regulated data exchange is to model organizations as a multi-agent system, where each organization has its own rules and policies to reason about the consequences of (not) exchanging data. Furthermore, organizations are able to argue with other organizations whether data can be exchanged. Thus, the balance of goals is not predetermined as in Buchanan et al.'s approach, but is found in the interaction between organizations. This research will investigate how the research from the AI & Law, argumentation and multi-agent system fields can be further developed, combined and implemented to support regulated data exchange in a serious problem domain.

1.2 Research aim

The introduction mentioned that regulated data exchange between organizations is often suboptimal because not enough data is exchanged legitimately and too much data is exchanged illegitimately. This problem results from organizations trying to achieve different goals and not knowing all the applicable regulations. The research aim of this thesis can be stated as follows.

> **Research aim**: *The research aim is to investigate how theories from multi-agent systems research, AI & Law research, and argumentation theory research can be further developed and applied in a realistic problem domain to provide a basis for automated support of organizations in promoting their goals in the context of regulated data exchange.*

1.2.1 Problem statement

Based on the research aim, the problem statement of this thesis can be formulated.

> **Problem statement**: *Is it possible to develop a realistic and implementable architecture for a multi-agent system that can provide a basis for automated support of regulated data exchange between organizations?*

Note that supporting regulated data exchange is interpreted as facilitating fewer unlawful and more lawful data exchanges.

1.2.2 Research method and research questions

The method used to answer the problem statement is as follows. The first step is describing and analyzing regulated data exchange in the Dutch police domain. The second step is investigating the suitability of a multi-agent system as a model for regulated data exchange, and specifying the requirements for the system. The third step is to model regulated data exchange as a formally specified multi-agent system. The fourth step is to describe the architecture and implementation of the multi-agent system. Each step has its own research questions which need to be answered.

Chosen domain with regulated data exchange

To determine how regulated data exchange can be supported, an application domain is needed. The Dutch police organization is a good example because it illustrates the problem of regulated data exchange. The method for acquiring knowledge about regulated data exchange in the Dutch police is to have interviews with police officers, and to make a study of the literature and the legal framework. Therefore, the first research question (RQ 1.) reads as follows.

RQ 1: How does the problem of regulated data exchange manifests itself in the Dutch police domain?

In order to support regulated data exchange in the Dutch police domain, a description of the legal framework is required. So, the first research question is subdivided into three subquestions.

RQ 1.a: What is the legal framework for the Dutch police domain?

In addition to the description of the legal framework, typical examples of interactions between members of organizations regarding data exchange need to be described.

What are the relevant interactions in this domain?

Recall that organizations must typically balance the goal to execute their tasks by exchanging data with the goal to protect their interests at the local level by not disclosing confidential data, while staying within the law. To promote the goals concerning the exchange of data, organizations can execute different types of local actions, and each chosen action can have possible undesired consequences on other goals. Therefore, to be able to choose which action is preferred, an analysis has to be made of the relations between actions, goals, and consequences on other goals.

What is the relation between goals and actions and possible consequences on other goals related to data exchange?

A multi-agent system as model for regulated data exchange

In order for the multi-agent system architecture to be a tenable model of regulated data exchange in the Dutch police practice, the requirements for the architecture need to be specified. Therefore, the second research question reads as follows.

RQ 2: What are the requirements for a multi-agent system suitable to support regulated data exchange in the Dutch police organization?

For agents to be able to regulate distributed data exchange, the agents must have knowledge of their domain.

RQ 2.a: Which kinds of knowledge do agents need to have?

Besides having knowledge of their domain, the agents also need to be able to reason with their knowledge of their domain.

RQ 2.b: Which reasoning capabilities do the agents need?

The agents must be able to engage in dialogues which model the typical interactions of the problem domain.

RQ 2.c: Which interaction types in this domain need to be supported by the multi-agent system?

Formalization of the multi-agent system

The requirements that result from the answers to question 2 must be formalized. This leads to the third research question.

RQ 3: How can the requirements for the multi-agent system be formalized?

The required reasoning capabilities of an agent need to be modeled. Here research question RQ 3 is subdivided in four subquestions.

How can the internal reasoning of an agent be formalized?

Knowledge must be represented in a way which is realistically implementable. This has as a consequence that the representation language must be executable by an

inference engine and needs to have enough expressive power to model the knowledge of the agents.

RQ 3.b: How can the knowledge of an agent be represented?

As discussed earlier, the agent interactions will be modeled with dialogue systems.

RQ 3.c: How can the agent interactions be represented?

In addition, the agents must be given policies for their behavior in the dialogues and the policies should be designed to further the agent's goals.

RQ 3.d: How can dialogue policies (or strategies) be defined that assist agents to balance their goals?

Agents also need internal policies to prescribe when the knowledge of an agent must be updated during a dialogue. For example if an agent with high authority states something, should the other agent update his knowledge base with the statement or just concede the statement?

RQ 3.e: Dialogues can result in knowledge updates. How can these changes in knowledge be modeled?

Specification and implementation of the architecture

To be able to implement the multi-agent system, its architecture needs to be specified.

RQ 4: What is the architecture of the multi-agent system for regulated data exchange between different organizations?

A requirement for the multi-agent system architecture is that it can be used to support regulated data exchange in similar domains. This leads to two subquestions.

RQ 4.a: How can the architecture of the multi-agent system be generalized to be applicable to similar domains?

In order to test the architecture with examples from practice, an implementation must be developed.

RQ 4.b: How can the architecture be implemented into a proof-of-concept application for a problem domain?

1.3 Thesis outline

This thesis is organized as follows. Chapter 1 presented the research aim and research questions for this research. Chapter 2 answers research question 1 by analyzing and describing regulated data exchange in the Dutch police domain. Chapter 3 answers research question 2 by specifying the requirements for a multi-agent system suitable to support regulated data exchange. Chapters 4 and 5 answer research question 3 by specifying how the requirement for a multi-agent system can be formalized. Chapter 4 will specify how the knowledge of an agent can represented and how the internal reasoning of an agent can be modeled. Furthermore, chapter 4 describes how the agent interactions can be formalized. Chapter 5 will specify the dialogue policies, which will assist agents to balance their goals. In addition, chapter 5 will specify the internal policies which prescribe when the knowledge of an agent must be updated during a dialogue. Chapter 6 answers research question 4 by describing the specification and implementation of the multi-agent system architecture that meets the requirements. Chapter 7 discusses related research on the topic of agents engaging in argumentation dialogues to regulate data exchange. Finally, chapter 8 presents the conclusions and ideas for further research. Note that this thesis is based on earlier work published in (Dijkstra et al. 2005; Dijkstra et al. 2006; Dijkstra et al. 2007). In Dijkstra et al. (2005; 2006) the initial outline of a multi-agent system architecture for regulated data exchange was presented. In Dijkstra et al. (2007) an early version of the formalization and implementation of the negotiation policies was specified.

Chapter 2

Regulated data exchange in the Dutch police domain

This chapter presents a description and analysis of regulated data exchange in the Dutch police organization[4]. First section 2.1 discusses the problem of regulated data exchange. Then section 2.2 describes the information systems involved in the data exchange and section 2.3 presents the legal framework. Section 2.4 discusses the approach to translating the applicable Dutch law into an intermediate representation, and describes the selected applicable rules and local rules. Section 2.5 presents three examples to illustrate the problem of regulated data exchange. Finally, section 2.6 analyzes patterns in the examples (in terms of goals, actions and possible consequences on other goals).

2.1 The problem of regulated data exchange

This section discusses the problem of regulated data exchange in the Dutch police domain. The Dutch police organization is divided into separate regional departments, which each operate in their own region[5]. Since severe crime is not bound by the regions of departments, departments often need data held by other departments in order to solve crime cases. The exchange of data between departments is governed by national and international privacy regulations and these regulations are implemented by local rules of the departments. With the exchange of data, organizations generally try to achieve two – sometimes conflicting – goals. On the one hand, they have the goal to execute their appointed police task by exchanging data. On the other hand, they have the goal to protect local

[4] Note that this chapter is based on earlier work published in Dijkstra et al. (2005; 2006).

[5] The Dutch police domain will restructure the 26 departments to one National Police consisting of 10 regional units. For more information, see http://www.rijksoverheid.nl/onderwerpen/politie/nationale-politie; accessed on November 3rd, 2011.

investigations and informants. In the regulations concerning data exchange, this balance of the two overall goals is also expressed. For example, some typical norms expressing the possible conflicting goals to exchange and to protect data are as follows.

> *It is obliged to exchange data from a police register with police officers, as far as they need this for the execution of the police task* (my translation of article 14, paragraph a of the Dutch Police Registers Act).

and

> *If necessary for the proper execution of the police task, the exchange from a severe-crime register is permitted to be refused* (my translation of article 13a, paragraph 3 of the Dutch Police Registers Act).

Ideally, dialogues between agents of different local departments guarantee that an optimal and legitimate balance of the goal to execute their appointed police task and the goal to protect local investigations and informants is found in the interaction. In consequence, when exchanging data with each other, police officers often have to communicate explicitly to make sure they serve local interests and at the same time conform to the regulations. Each department has a criminal investigations unit (*criminele inlichtingen eenheid*, CIE) for gathering intelligence data on severe crime. A typical interaction between CIEs is about the exchange of data acquired from informants[6]; about 80% of police data on severe crime in the examined departments is obtained from informants (Bonthuis 2003). Since informants often have some kind of involvement in the crime they supply data about or are afraid of possible retaliations, criminal investigations units are extremely cautious about the exchange of this kind of data, even if the exchange of data is obliged or allowed by law. If a crime suspect confronted with data obtained from informants finds out who supplied the data, the safety of the informant and the continuity of the investigation performed by the department that supplied the data is endangered. Therefore, in most cases the criminal investigations unit that 'runs' the informant will not be willing to exchange the data unless the receiving criminal investigations unit offers certain guarantees.

The activities of CIEs can be divided into three groups: collecting data from informants, analyzing severe-crime data, and exchanging severe-crime data (Werkgroep CIE 2003). Since the chosen subject of this research is regulated data exchange, the focus is on exchanging severe-crime data. The exchange of severe-crime data can be further broken down into four steps (Subwerkgroep Criminele Inlichtingen Eenheden 2002). The first step is deciding whether the request from

[6] An informant is a person who, other than being a witness or a suspect, supplies data to the police about criminal events.

the other party can be accepted. During the first step the authorization and the reason for the request from the requesting party are checked. The second step is selecting relevant data. In the second step, relevant data is searched in the local severe-crime register and *MRO*s[7]. If needed, informants are contacted to give updates on the requested data. The third step is the actual exchange of data. In the third step, data is joined and organized into a chosen predefined format. The fourth step is logging the exchange. The logging step records to whom what data was supplied and under which agreements or conditions.

2.2 The RBS information system

This section discusses how the data exchange process described above occurs in practice. Within the Dutch police domain, there are many information systems for specific purposes. Most departments use the 'Recherche Basis Systeem' (RBS) information system as their tool for entering and searching information. RBS is an information system specifically developed for the Dutch police practice and is designed for capturing, analyzing, authorizing, and categorizing information (Van der Velde 2003). In RBS, data is divided into separate *registrations*. A registration is a digital version of a register and is only available to authorized people. A registration consists of *mutations*. A mutation is a new unit of data added to a registration, not a change of existing data. The rationale behind this is that data in a mutation is *set definitive* (that is, made read only) after 14 days; this ensures that the data has not been tampered with[2]. A *mutation* has a text source and metadata criteria. An example of a metadata criterion used to enable data exchange is called *data usage code,* which is added to every mutation. The data usage codes determine whether and how the data can be used and are further described in subsection 2.4.3. Besides the data usage codes, there is also metadata on the reliability of data. The reliability metadata is divided into two parts, the reliability of the supplier of the data and the reliability of the data itself. For example, if an informant was trustworthy in the past, data originating from him will have more value and consequently the *data supplier reliability* will be evaluated as reliable.

In the Dutch police domain, the following metadata values are used for data reliability (Bonthuis 2003).

Data supplier reliability values

[7] An investigation for which it is possible to obtain an arrest warrant and which is expected to take more than one week is called an MRO. MRO stands for '*Melding Recherche Onderzoek*', which can be translated as 'notification for a police investigation'.

A = *reliable*

B = *usually reliable*

C = *less or not reliable*

X = *cannot be determined*

Data reliability values

1 = *observed directly by the data supplier*

2 = *observed by someone who told it directly to the data supplier*

3 = *the data supplier heard it indirectly*

Note that in most cases the data supplier is an informant.

The process of adding of criminal intelligence data into RBS can be illustrated as follows. Assume that Salvatore is an informant and tells to a CIE *runner*[8] about the location where a large quantity of drugs belonging to Soprano can be found. First the CIE runner makes a report of the conversation. This report is then *refined,* meaning that the data is verified, evaluated, and if possible supplemented with data from other sources. It is also checked whether the report can be legally stored in the local severe-crime registration. After the refinement and legal checks, the report will be added in RBS as a *mutation* in the severe-crime register of the local police department. Now consider what the possible consequences are if this intelligence data is used to solve a crime investigation on Soprano. If Salvatore is the only person (besides Soprano) who is in a position to know about Soprano's drugs location, then Salvatore can be in serious trouble. A possible consequence is that Soprano orders to kill the informant. In this scenario a crime investigation on Soprano is completed. However, the informant (who probably is also used for other investigations) is lost. A second possible consequence is that the informant is not killed but loses his trust in the CIE runner and stops giving criminal intelligence data. These example scenarios also illustrate how crime investigations and informants are tightly coupled and why severe-crime data needs to be protected.

2.3 Legal framework

This section discusses the applicable legal framework. First some legal background information about the Dutch police organization is presented. Then the intended free flow of information versus the refusals in practice is discussed. Next the concept 'police task' is discussed and finally the police registers are described.

[8] Runners are CIE officers who are in direct contact with the informants.

2.3.1 Legal background information about Dutch police organization

The two overall tasks of the Dutch police organization are assistance to the general public and maintaining law and order (article 2 of the law for maintenance of public order, *Politiewet*). One of the subtasks of maintenance of law and order is tracing criminals, where the main goal is gathering evidence in criminal cases. A second subtask is the general intelligence process, in which data is collected about crimes which are committed or intended. General intelligence data is used to decide which crime investigations will be started. In the Dutch police organization, the criminal intelligence units (CIEs) are responsible for the gathering of intelligence data for the police task as far it is related to severe crime. Every police department has an autonomous CIE and there is one national coordinating CIE.

In the Netherlands the overall encompassing law concerned with the processing of personal data is called the Personal Data Protection Act (*Wet bescherming persoonsgegevens*, Wbp). The Wbp states that processing of criminal personal data in the context of criminal law is forbidden (article 16, Wbp). However, for the processing of data by the police an exception is made (article 22, paragraph 1, Wbp).

While the primary goal of the Wbp is to protect citizens against privacy infringement, the primary goal for the police is the execution of their appointed police task. For example, the obligation from the Wbp to inform people involved in the processing of data can seriously damage a police investigation and interfere with the execution their appointed police task (De Vey Mestdagh 2008). Because of the special nature of police intelligence, specific laws have been issued and the Wbp is not applicable in this case. For the exchange of severe-crime data between CIEs, the Police Registers Act (*Wet politieregisters,* Wpolr) was applicable at the time of this research, which is further detailed in the Police Registers Decree (*Besluit politieregisters,* Bpolr) and the CIE regulation (*regeling CIE*). Since 1 January 2008, both the Wpolr and the Bpolr have been replaced by the Police Data Act (*Wet politiegegevens*, Wpg) and the Police Data Decree (*Besluit politiegegevens*, Bpg). During the specification of the model and implementation of the proof of concept application, the Wpolr and Bpolr were applicable. Therefore, the present research uses the Wpolr and the Bpolr. However, the results of this research can be generalized to the present legal context in the Dutch police domain because both legal frameworks are similar with respect to their contents. The main difference between the two legal frameworks is the change of focus from individual registrations to individual data. However, the change of focus does not change the conclusions of this research since registrations consist of data. The previous and current legal frameworks are both characterized by the protection of data and informants and leave room for local decisions. For example, Wpg article 27,

paragraph 1, subparagraph b, allows refusing a request for data for the important interests of third parties; which refers to the protection of informants and infiltrators[9]. This ground for refusal was also expressed in Bpolr article 11, paragraph 1, subparagraph a.

2.3.2 Intended free flow versus refusals in practice

The legislator intended a free-flow of data within parts of the Dutch police organization (Ruth and Schreuders 2000; Schreuders and Wel 2005) and has created many rules which obligate the distribution of data. However, to enable data protection, the legislator also allows grounds for refusal as a legal way to refuse the exchange of data. It was the legislator's goal that in specific cases of data protection, the relevant interests are weighed against each other. However, my interviews with police officers revealed that in practice the behavior is almost the opposite of free-flow; the grounds for refusal are more used as default instead of using it in special cases with serious deliberation (see also Kielman 2010, page 133). The large number of refusals has as consequence that only a part of the permitted data exchanges occur in the Dutch police domain. The present research is aimed at creating fewer refusals by which regulated data exchange between organizations can be supported.

2.3.3 The concept 'police task'

The concept 'police task' is central both in the Police Registers Act (Wpolr) and the Police Registers Decree (Bpolr), see Ruth and Schreuders (2000). First it determines which types of data can be stored. Second, it determines whether it is obliged or permitted to refuse to exchange data. Finally, 'police task' is used as ground for applicability for the Wpolr and Bpolr. If the Wpolr and Bpolr are not applicable, the Wbp is applicable and therefore also its distribution regime.

The main rule promoting the exchange of data is Wpolr article 14, paragraph a, which states that it is obliged to exchange data if the other party needs it for the execution of the police task; here the police task is used as a ground to promote the exchange of data. The main rule limiting the exchange of data from severe-crime registers is Wpolr article 13a, paragraph 3, which states that the exchange of data can be refused if necessary for the proper execution of the police task; here the police task is used as a ground to refuse the exchange of data and therefore limiting data exchange. If the concept of 'police task' is used as a ground for refusal, it must be argued for in every particular case. The two general demands which are relevant are proportionality and subsidiarity. As described in Mommers et al. (2007), the concept of 'necessity' in Wpolr article 13a paragraph 3 refers to the demands of proportionality and subsidiarity. In the context of this article, proportionality is an

[9] *Kamerstukken* II, 2005-06, 30 327, nr. 3, p. 85.

assessment whether the (possible) negative consequences of an action are in proportion to the intended positive consequences. For example, the consequences of revealing the identity an informant for finding the location of a drug house are not severe compared to the benefits of protecting the informant if he is providing valuable data about a serial killer. The demand of subsidiarity entails that always the least drastic still feasible measure should be taken. In this case, it should be examined if there is a possibility to exchange the data in a way that has no or less impact on the execution of the police task. For example, instead of refusing to exchange data for finding the location of a drug house (because it will reveal the identity of the informant supplying data about a serial killer), the data could be exchanged for analysis purposes only. If the data is used for analysis purposes only, the data is analyzed to find general patterns. An example of a pattern is the typical location of a drug house, which can be used for starting new investigations. Grounds for refusal can be deeply intertwined. For example, the protection of informants is a part of the police task and part of privacy protection. Informants will not cooperate if their privacy is not protected and if they do not cooperate, the execution of the police task is impeded (Ruth and Schreuders 2000).

2.3.4 Police registers

Data needed to fulfill the police task is stored in registers. A register is defined as a coherent collection of personal data related to several persons that has been created to serve the execution of the police task (my translation of article 1, paragraph c, Wpolr). Every police region creates and administrates their police registers. A severe-crime register is a police register specifically for storing severe-crime data.

Within the police organization, it is only allowed to create registers for a specific purpose, for example, the tracing of perpetrators of criminal offences. The tie-to-purpose principle states that storage of data is only allowed if necessary for the purpose of the register. Deviations from the tie-to-purpose principle are allowed if necessary, for example the tracing of crimes (College bescherming persoonsgegevens 2004). The police organization also needs to design specific privacy regulations for every police register (article 9, Wpolr). Privacy regulations for police registers are mandatory and state the purpose and the administrator of the register. In addition, a privacy regulation states who have access to a register, which is partly done by assigning roles. The register administrator has to designate a specific list of people who are authorized to have access. In order to facilitate uniformity of the same type of registers in different departments, article 12, paragraph 1 of the Wpolr allows the creation of *model regulations.* A model regulation is a blueprint for privacy regulations for police registers. If a register administrator creates a new register based on a model regulation, he only has to inform the Dutch Data Protection Authority (*College bescherming persoonsgegevens*, CBP).

2.4 Approach to translating applicable Dutch law into an English intermediate representation

In order to create a multi-agent system architecture and a proof of concept application to demonstrate how the regulated data exchange between different organizations can be optimized, the applicable rules need to be formalized. In this section the selected applicable rules are rewritten into an intermediate representation which will be formalized in later chapters. The reasons for rewriting the applicable rules into an intermediate representation are that an intermediate representation (1) creates a link between the formalization and the applicable rules, (2) is independent of the implementation, and (3) allows for easier validation (Bench-Capon and Coenen 1992; De Vey Mestdagh 1997). The intermediate representation rules are written from the perspective of a CIE agent who has to decide whether it is obliged, allowed, forbidden or it is a violation of his own interests to exchange intelligence data. First the approach to translating, selecting, interpreting, and rewriting the applicable regulations is described. Then the underlying assumptions used for selecting and translating the applicable laws into the intermediate representation are specified (2.4.1). Finally, the selected applicable laws (2.4.2) and (2.4.3) local rules are translated into an English intermediate representation.

2.4.1 Assumptions

This subsection presents the assumptions for creating an architecture of a multi-agent system for regulated data exchange in the Dutch police domain. In order to keep the model of regulated data exchange in the Dutch police domain manageable, the following 11 assumptions are applied in the selection of what is relevant in the applicable regulations.

- Data is stored severe-crime registers.
- The distributor of data is an authorized CIE officer who is a member of the local criminal investigation unit and has permission to have full access (i.e., register administrator role) to the local severe-crime register.
- The requester of data is an authorized CIE officer from another criminal investigation unit and has permission to receive data from other severe-crime registers.
- CIE officers do not exchange intelligence data outside criminal investigation units. Note that the condition stating that it is forbidden to pass on the exchanged intelligence data to other departments or criminal investigation units is not modeled to allow easier implementation. Of course, the model can be extended to allow exchanging data outside criminal investigation units.

- The goals of CIE officers are to execute the police task and to protect local investigations and informants.
- CIE officers trust each other.
- Criminal investigation units can have different interpretations of the legal framework and therefore can have different local rules[10].
- Criminal investigation units do not hide data[11], that is, if they have data but they are not willing to supply it, then they tell this to each other. In practice, this assumption is also made by CIE officers[12].
- Within a criminal investigation unit, all CIE officers have the same 'attitude' (see section 2.6) towards data exchange.
- CIE officers only refuse to exchange data for the proper execution of the police task.
- CIE officers only request data for the proper execution of the police task. Note that the model incorporates other norms for regulated data exchange between criminal investigation units, and the model can be extended to incorporate this norm.

2.4.2 Selected applicable articles

In this subsection the selected applicable three articles are translated into an English intermediate representation. The first step of the approach is selecting the original Dutch articles from applicable regulations that help to decide whether severe-crime data can be exchanged with other CIE agents. The original Dutch articles are presented first to enable validation of the translation. The second step is translating the selected articles into English; next in the third step the relevant excerpts from the articles are selected (relevance is based on the assumptions described above). In the fourth step, the selection and interpretation of relevant excerpts are explained and motivated. In the fifth step, the excerpts are rewritten into an intermediate representation.

1: Wpolr art. 14 par. a (main rule for exchanging data with CIEs)

Step 1: Original Dutch article

[10] Note that local rules can also have different interpretations.
[11] Note that Teepe does not make this assumption in his thesis (Teepe 2007).
[12] Based on the interview with Doetse Huizinga, conducted with Hugo Kielman and Wouter Koelewijn on 30-3-2005 in Apeldoorn.

Uit een politieregister worden gegevens verstrekt aan:

a. ambtenaren van politie, voor zover zij deze behoeven voor de vervulling van de politietaak en zij niet zijn aangesteld voor de uitvoering van technische, administratieve en andere taken ten dienste van de politie en aan ambtenaren die krachtens artikel 13c, vijfde lid, zijn aangewezen, voor zover zij deze behoeven ter opsporing van strafbare feiten;

Step 2: My translation

It is obliged to exchange data from a police register with:

a. police officers, as far as they need this for the execution of the police task and they have not been appointed for the execution of technical, administrative and other task as service to the police and to officials who have been assigned by article 13c, paragraph 5, as far as they need this for the investigation of criminal offences;

Step 3: Relevant excerpts from translated article

It is obliged to exchange data from a police register with police officers, as far as they need this for the execution of the police task.

Step 4: Interpretation and explanation

This article contains the general norm for exchanging data and is also applicable for CIE agents, since they are police officers who have not been appointed for the execution of technical, administrative and other task as service to the police. Note that the section 'as far as they need it for the execution of the police task' has two implications. The first implication is 'exchange data if the other police officer needs it'. The second implication is 'do not exchange if the other police officer does not need it'. An assumption is that CIE officers only request data for the proper execution of the police task. For that reason, Wpolr article 14, paragraph a, the second implication is not applicable and is therefore not represented. Note further that the general police register is substituted by a severe-crime register, which are the registers used by CIEs when they exchange data.

Step 5: Rewrite of excerpts into intermediate representation

(Wpolr14.a implication 1)

IF *a CIE agent needs data for the execution of the police task,*
THEN

it is obliged to exchange the data from the severe-crime register with the CIE agent.

2: Wpolr art. 13a par. 3 (main rule for refusing the exchange of severe-crime data with CIEs)

Step 1: Original Dutch article

Indien dit noodzakelijk is voor een goede uitvoering van de politietaak, kan de verstrekking ingevolge de artikelen 14 en 15, eerste lid, onder b, c en d en f, uit een register zware criminaliteit worden geweigerd dan wel aan beperkende voorwaarden wat betreft het verdere gebruik worden onderworpen.

Step 2: My translation

If necessary for the proper execution of the police task, the exchange in accordance with articles 14 and 15 first paragraph under b, c, and d, from a severe-crime register is permitted to be refused, or carried out under restricting conditions regarding to further use.

Step 3: Relevant excerpts from translated article

If necessary for the proper execution of the police task, the exchange from a severe-crime register is permitted to be refused or carried out under restricting conditions regarding to further use.

Step 4: Interpretation and explanation

This article refers to article 14 Wpolr, which is the general norm for exchanging data from police registers, and contains the norm for refusing or restricting the exchange of severe-crime data. The translation uses ‘is permitted to’ instead of ‘can’ to clearly specify the permission to refuse to exchange severe-crime data or to state conditions for further use. References to severe-crime registers are left implicit in the intermediate representation, which is based on the assumption that all data is stored in severe-crime registers.

Step 5: Rewrite of excerpts into intermediate representation

(Wpolr13a.3 part 1)

IF *a refusal to exchange data is necessary for the proper execution of the police task,*
THEN
it is permitted to refuse to exchange data.

(Wpolr13a.3 part 2)

IF a refusal to exchange data is necessary for the proper execution of the police task,
THEN
it is permitted to exchange data with restricting conditions regarding to further use.

3: Bpolr art. 11 (grounds for refusal)

Step 1: Original Dutch article

1. Een beheerder is bevoegd verstrekking van gegevens uit een politieregister ingevolge de artikelen 14 en 15, eerste lid, onder b en c, van de wet te weigeren indien:

a. het gegevens betreft omtrent personen die aan de politie informatie hebben verstrekt omtrent door anderen gepleegde of te plegen strafbare feiten;

b. het gegevens uit een register betreft waarbij, mede gelet op de bijzondere aard van het register, in geval van verstrekking direct gevaar voor de geregistreerde of voor derden zou zijn te duchten.

3. Het eerste en tweede lid vinden slechts toepassing indien dit noodzakelijk is voor de goede uitvoering van de politietaak. Bij de verstrekking van de daar bedoelde gegevens kunnen beperkingen aan het gebruik van de gegevens worden opgelegd.

Step 2: My translation

1. An administrator is permitted to refuse to exchange data in accordance with articles 14 and 15 paragraph 1, subparagraphs b and c, of the law if:

a. it concerns data about people who have supplied data to the police concerning crimes which were or are going to be committed;

b. it concerns data from a register of which, given the special nature of the register, in case of exchanging data, creates immediate danger for the registered person or third parties.

3. The first and second paragraph are not applicable when not necessary for the proper execution of the police task. During the exchange of the data meant above, it is permitted to enforce restrictions on the usage of data.

Step 3: Relevant excerpts from translated article

1. An administrator is permitted to refuse to exchange data in accordance with articles 14 of the law if:

a. it concerns data about people who have supplied data to the police concerning crimes which were or are going to be committed;

b. it concerns data from a register of which, given the special nature of the register, in case of exchanging data, creates immediate danger for the registered person or third parties.

3. The first paragraph is not applicable when not necessary for the proper execution of the police task. During the exchange of the data meant above, it is permitted to enforce restrictions on the usage of data.

Step 4: Interpretation and explanation

Since the data exchange between CIEs is fully covered by article 14 of the Wpolr, the part 'and 15 paragraph 1, subparagraphs b and c' can be omitted in the relevant excerpts. The excerpt 'people who have supplied data to the police' from subparagraph a, paragraph 1, refers to informants. Therefore, in the intermediate representation the term informants is used.

Step 5: Rewrite of excerpts into intermediate representation

(Bpolr 11.1. a)

IF *the data about informants concerns crimes which were or are going to be committed*

THEN

an administrator is permitted to refuse the exchange of data.

(Bpolr 11.1. b)

IF *the data originates from a register of which, given the special nature of the register, in case of exchanging data, creates immediate danger for the registered person or third parties*

THEN

an administrator is permitted to refuse the exchange of data.

(Bpolr 11.3 sentence 1)

IF *a refusal to exchange data is not necessary for the proper execution of the police task,*

THEN

(Bpolr 11.1) is not applicable.

(Bpolr 11.3 sentence 2)

IF *data is exchanged*

THEN

it is permitted to enforce restrictions on the usage of data.

2.4.3 Selected applicable local rules

In this subsection a selection of local rules is translated, interpreted and rewritten into an intermediate format. The local rules were acquired using interviews with police officers of the Dutch police[2] (Bonthuis 2003). The local rules are based on local interpretations which are allowed within the given discretionary power to the CIE units. The interpretations are in turn based upon the 'attitude' of a department organization. For example, a criminal investigation unit with a cooperative attitude interprets the applicable regulations in a way that leaves more room for exchanging data. An example of different local interpretations is that some departments exchange severe-crime data only after permission of the CIE-chief, while other departments exchange data without permission and tell the CIE-chief afterwards (in case of emergencies). This can be considered as two different interpretations of article 4, paragraph 1, subparagraph b of the CIE-regulation, which states that CIEs are obliged to conform to the Model Regulation Severe Crime (*modelreglement register zware criminaliteit*, MRZC) and article 4 of MRZC which states that the register manager has authority on the register.

The first step is selecting the original local rules that help to decide whether severe-crime data can be exchanged with other CIE agents. The second step is translating the local rules into English. In the third step, the local rules are interpreted and explained. In the fourth step, the local rules are rewritten into an intermediate representation.

The local rules are organized as follows. First the data usage codes are rewritten (A), then rules stating grounds for refusal are described (B), and finally rules stating exceptional cases are presented (C).

A: Data usage codes

The data usage codes determine the allowed usage of data (see appendix 1 of the CIE regulation). The usage codes are applied within the whole police organization. The CIE chief (*Groepschef*) decides, in consultation with the CIE officers who are in direct contact with the informants (*runners*), which data usage codes will be used. Below, the data usage codes are translated and rewritten into an intermediate representation.

Step 1: Original Dutch data usage codes

11 = *operationeel te gebruiken*

01 = *alleen gebruiken na overleg met afzender*

00 = *informatie met zware beperkingen voor gebruik*

200 = *kan niet operationeel gebruikt worden, maar kan onder bepaalde voorwaarden wel voor coördinatie- en analysedoeleinden worden gebruikt. + informatie met verhoogd afbreukrisico*

300 = *kan niet operationeel gebruikt worden, maar kan onder bepaalde voorwaarden wel voor coördinatie- en analysedoeleinden worden gebruikt. + informatie met bronbeschermingsbelangen*

Step 2: My translation

11 = *ready for operational usage*

01 = *only permitted to use after consultation with the sender of the data*

00 = *data with severe restrictions on usage*

200 = *forbidden to use for operational usage, but only permitted to use under certain conditions for coordination and analysis purposes. + data with risk factor*

300 = *forbidden to use for operational usage, but only permitted to use under certain conditions for coordination and analysis purposes. + data with source protection*

Step 3: Interpretation and explanation

The data usage codes are an implementation of art. 11, par. 3. of the Bpolr 'it is permitted to enforce restrictions on the usage of data' and 11.1.a of the Bpolr 'An administrator is permitted to refuse to exchange data in accordance with articles 14 and 15 paragraph 1, subparagraphs b and c, of the law if: a. it concerns data from people who have supplied data to the police concerning crimes which were or are going to be committed'.

Although all criminal investigation units use the same data usage codes, different interpretations create differences in the application of those codes. The data usage codes are tentative and leave discretionary room; this can be observed in their formulations. For example, usage code *01* states that data can only be used after consultation with the sender. How this consultation is done and when it is sufficient to start exchanging data is a local decision. An example of different local interpretations is that some regional departments with an overcautious attitude apply the data usage code *00* (data with severe restrictions on usage) often to make sure that their data is not easily shared. Furthermore some CIE units almost never want to exchange, while other units can be persuaded to exchange data with access code 00^2. A second example of different local interpretations is that in some departments, runners cannot see their own created data with data usage code 00^2. Here an example of the difference between the intended free flow and the overused grounds for refusal in practice can be observed. Only non-protected data which has data usage code *11* gives room for a free flow of data. The formulation 'operational usage' indicates that the intelligence data can also be used and exchanged freely with authorized police officers other than CIE officers. Bpolr 11.1 distinguishes two grounds for refusal: the protection of informants (Bpolr 11.1.a) and the risk of revealing the registered people or third parties (Bpolr 11.1.b). Data usage code *300* is an interpretation of Bpolr article 11, paragraph 1, subparagraph a, where the protection of informants is called "source protection". Data usage code *200* is an interpretation of Bpolr article 11, paragraph 1, subparagraph b, where the risk of revealing the registered people or third parties is called a "risk factor".

Exchanging data labeled with 'source protection' or with a 'risk factor' can have undesired consequences on the protection of local investigations: when an informant is at risk then also the investigation which uses the informant as a source can be at risk, and when the registered person or third party (for example an anonymous witness) is at risk then also the investigation can be at risk. Risks arise when the exchange of data has undesired consequences on the police or on the sources of intelligence data. Examples of undesired consequences are: data becoming useless, harming the police reputation when an anonymous witness is revealed, and harming the relationship with informants when an informant is

revealed. The risk that an informant or third party will be killed is the most serious risk factor (Kielman 2010, page 43; Koelewijn 2009, page 104).

Data usage code *00* does not specify why the data is protected. Therefore in practice, a CIE agent will look into the mutation text source (see section 2.2) or ask the creator of the text to give reasons for the *00* usage code if a discussion with other CIE agents comes up. Data usage codes *200* and *300* were added later to the already existing code *00* to make it more clear why the data is protected (i.e., to be used as a ground for refusal).

Let us return to the example of section 2.2, where Salvatore is an informant and tells a CIE runner where a large quantity of drugs belonging to Soprano can be found, to illustrate how the data usage codes are applied. Assume that the data was already disclosed by another informant and Soprano was already caught by another department for this crime, then the mutation could probably have *11* as usage code. If Salvatore is the only source and he could be important in other crime cases, the mutation could be evaluated as *300*. If the runner knows other informants who also know about where a large quantity of drugs belonging to Soprano can be found, he might be willing to exchange the data and therefore could label the mutation as *01*.

Step 4: Rewrite of excerpts into intermediate representation

Data usage code 11

IF *data has usage code 11*
THEN
it is obliged to exchange the data.

Data usage code 01

IF *data has usage code 01*
THEN
it is permitted to exchange the data only after consultation with the sender.

Data usage code 00

IF *data has usage code 00*
THEN
the data has severe restrictions on usage.

Data usage code 200 forbidden

IF data has usage code 200
THEN
it is forbidden to use the data for operational usage.

Data usage code 200 coordination and analysis

IF data has usage code 200
THEN
it is permitted to use the data only under certain conditions for coordination and analysis purposes.

Data usage code 200 risk factor

IF data has usage code 200
THEN
the data is a risk factor.

Risk factor then forbidden to exchange

IF data has a risk factor
THEN
it is forbidden to exchange the data.

Risk factor and analysis permitted to exchange

IF data has a risk factor and is used for analysis purposes
THEN
it is permitted to exchange the data.

Data usage code 300 forbidden

IF data has usage code 300
THEN
it is forbidden to use the data for operational usage.

Data usage code 300 coordination and analysis

IF data has usage code 300
THEN

it is permitted use the data only under certain conditions for coordination and analysis purposes.

Data usage code 300 source protection

IF data has usage code 300
THEN
the data has source protection.

Source protection forbidden to exchange data

IF data has a source protection
THEN
it is forbidden to exchange the data.

Source protection and analysis permitted to exchange data

IF data has a source protection and is used for analysis purposes
THEN
it is permitted to exchange the data.

B: Common grounds for refusal

Beside the data usage codes *00*, *200*, and *300* other more explicit grounds for refusal can be used. Examples of more explicit grounds for refusal are[2]: when the data is used for an internal investigation, when the data is no more up to date, or when the data is traceable to informants. Traceability to informants is partly handled by usage code *300*; the difference is that data with usage code *300* is directly about informants, while data labeled traceability can indirectly lead to informants. By giving more explicit grounds for refusals, agents have more means to engage in dialogues which help to optimize regulated data exchange. Below, the rules expressing grounds for refusals are described and rewritten into an intermediate format.

B1: Ground for refusal: traceability of informant

Step 1:Original Dutch rule

Gegevens die herleidbaar zijn naar een informant hebben bronbescherming.

Step 2: My translation

Data which are traceable to an informant has source protection.

Step 3: Interpretation and explanation

This domain rule is based on interviews with CIE officers in the Dutch police domain[2]. The estimation that the data can be used to trace an informant is an important reason for source protection. A traceability assessment is dependent on the number of informants who have given the data and the level of detail of the data. If data is too specific then the suspect could trace the person who has given the data. For example, if Adriana is the only one who knows that Christopher Moltisanti has heroin in the lower drawer of his bedside table then, when this information is used, Christopher will know that Adriana is an informant.

Step 4: Rewrite of local rule into intermediate representation

Traceable source protection

IF *data is traceable to an informant*
THEN
the data originating from the informant has source protection.

B2: Ground for refusal: internal investigation

Step 1: Original Dutch rule

Het is verboden om gegevens die gebruikt worden voor een intern onderzoek over een collega te verstrekken.

Step2: My translation

It is forbidden to exchange data used for internal investigation about a colleague.

Step 3: Interpretation and explanation

An internal investigation can be conducted on a colleague police officer or a criminal investigation unit suspected of leaking data back to criminals. When a police officer or a criminal investigation unit leaks data back to criminals, local investigations can be at risk. Therefore this can be regarded as an interpretation of a risk factor of art 11. par. 1 subpar. b of the Bpolr. Recall that a risk factor is defined as the risk of revealing the registered people or third parties (see interpretation and explanation of data usage code *200* on page 27).

Step 4: Rewrite of local rule into intermediate representation

Colleague then risk factor

IF *data is used for internal investigation about a colleague*
THEN
the data has a risk factor.

B3: Ground for refusal: older than one year

Step 1: Original Dutch rule

Gegevens ouder dan 1 jaar worden niet meer verstrekt.

Step 2: My translation

It is forbidden to exchange data older than one year.

Step 3: Interpretation and explanation

This local rule is based on interviews with CIE officers in the Dutch police domain[2]. In the Dutch police practice, data older than one year is considered as not up to date. For that reason it is forbidden to exchange data older than one year. Informants first need to be consulted to revalidate the data before it can be used again. In the context of exchanging data this rule is more stringent than Wpolr art. 13a par. 8. This article states that data must be deleted when older than five years and therefore permits to exchange data not older than five years.

Step 4: Rewrite of local rule into intermediate representation

Older than one year forbidden to exchange data

IF *data is older than one year*
THEN
it is forbidden to exchange the data.

C: Exceptional situations: National importance, attempted assault on the queen, multiple child-murder

The general norm for exchanging data is Wpolr art. 14 par. a. For exchanging severe-crime data, article 13a, paragraph 3Wpolr specifies which grounds for refusals can be used for exceptions for Wpolr art. 14 par. a. However, based on interviews with police officers[2], those exceptions in turn can also have exceptions.

Those exceptions are called exceptional situations and specify in which cases exceptions can be set aside.

Step 1: Original Dutch rule

> *In uitzonderlijke gevallen (zoals nationaal belang, een aanslag op de koningin of een meervoudige kindermoord) kan de bescherming van gegevens opzij worden gezet.*

Step 2: My translation

> *In exceptional cases (such as national importance, an attempt to assault the queen or a multiple child-murder) it is permitted to set aside the protection of data.*

Step 3: Interpretation and explanation

In exceptional cases the rules forbidding the exchange of data can be set aside. The risk of jeopardizing the police investigations or informants then has to be accepted by the CIE agent who owns the data. Examples of exceptional cases are: national importance and an attempted assault on the queen[2]. Of course other examples of exceptional situations exist.

Step 4: Rewrite of local rule into intermediate representation

Exceptional situations

IF *National importance*
OR
Attempted assault on the queen
THEN
an exceptional situation occurs.

Exceptional situation set aside protection

IF *an exceptional situation occurs*
THEN
then the rules forbidding the exchange of data are not applicable.

2.5 Examples of dialogues about regulated data exchange

In this section examples of regulated data exchange from the Dutch police practice are presented. For this research the examples are utilized to clarify the requirements for developing a multi-agent system (MAS) architecture for regulated data exchange. The examples will also be used to illustrate the communication protocol in Chapter 4 and the policies of Chapter 5. The chosen examples are based on data gathered from interviews[2]. Police officers have confirmed that the chosen scenarios are exemplary dialogues about regulated data exchange.

The exchange of severe-crime intelligence data between CIEs can be broken down into three classes. The first class is where requests for data are always granted directly. The second class is where data is never exchanged. The third class is the most interesting, it is the class where agents try to persuade or negotiate with each other in order to exchange data. The selected examples fall in the third class

Example 1 source protection & analysis purposes

In this example, agent *p* of criminal investigation unit *P* requests the responding agent *o* of criminal investigation unit *O* to give him intelligence data about Adriana.

p_1: Tell me all you know about Adriana.

After searching and finding data about Adriana in his severe-crime register, the responding agent o sees that the data has source protection since the data is traceable to his informant Adriana. Furthermore, agent o infers that informant Adriana has no protection when the data is exchanged. This is a violation of the interests of agent o since his goal to protect his informant Adriana conflicts with the conclusion that Adriana is not protected when the data about Adriana is exchanged. As a possible way to enable data exchange, agent o tries to find a condition under which the data can be exchanged. This is possible and agent o makes an offer to exchange the data on the condition that it only can be used for analysis purposes.

o_2: I will give you this data about Adriana under the following condition: the exchanged data may only be used for analysis purposes.

The requesting agent deliberates whether he can accept the condition with the offer. This is possible, since his query was made for analysis purposes

and he will use that data for analysis purposes only. Therefore, the requesting agent p accepts the offer

p_3: I agree with this condition.

Requesting agent p agrees with the condition and the responding agent sends him the requested intelligence data about Adriana.

Example 2 terrorist assault on the queen

In this scenario, CIE agent *p* of criminal investigation unit *P* has anonymously received a tip-off and asks agent *o* of criminal investigation unit *O* about a terrorist group in Amsterdam which is possibly planning an assault on the queen. Initially the responding agent rejects his request. However, the requesting agent persuades him to exchange the requested data

p_1: Give me all data about a possible terrorist attack.

At this moment, the responding agent o internally deliberates whether it is obliged, forbidden or whether there is violation of his interests to exchange the found data about a possible terrorist attack. The responding agent concludes that it is forbidden to exchange the data about explosive materials, since the data is older than one year. Therefore, the responding agent rejects the request.

o_2: I will not give you this data.

Since requesting agent p wants to gather as much data as possible, he will always ask for the reason after a rejection in order to try to create a counterargument. Hence, the requesting agent p asks why.

p_3: Why don't you give me that data?

Since all agents need to cooperate, for the proper execution of the police task, the responding agent o gives the reason for his rejection.

o_4: Because it is forbidden to exchange data about a possible terrorist attack.

The requesting agent p asks for a reason why it is forbidden to exchange data.

p_5: Why is it forbidden to exchange that data?

The responding agent o gives as reason that the action to exchange the data is forbidden since the data is older than one year.

o_6: Because the data is older than one year.

The requesting agent p agrees that generally data cannot be exchanged if the data is older than one year. However, in this case the data must be exchanged because of an exceptional situation.

p_7: You may be right in general but in this case it not forbidden to exchange the data since this is an exceptional situation.

The responding agent o asks for a reason why it is an exceptional situation.

o_8: Why is this an exceptional situation?

Because the requesting agent wants to convince the responding agent, he gives the attempted assault on the queen as a ground for the exceptional situation.

p_9: Because there is an attempted assault on the queen.

Responding agent o reasons that an attempted assault on the queen is more significant than not exchanging data older than one year. Thus he agrees that in this exceptional case the data must be exchanged.

o_{10} Ok, I concede that in this case it is not forbidden to exchange the data, so I retract that it is forbidden to exchange the data; I will offer you the requested data.

The requesting agent agrees with the offer.

p_{11}: I accept the offer.

The dialogue is completed and the outcome of the dialogue is that the data is sent to the requesting agent.

The initial refusal is based on article 13a, paragraph 3 of the Wpolr. However, in practice, in exceptional cases the rules forbidding the exchange of data can be set aside (see subsection 2.4.3 on page 31).

Example 3 investigation protection & national importance

In this example, agent *p* of criminal investigation unit *P* requests data about trading in explosive materials from agent *o* of criminal investigation unit *O*. Agent *o* is of a criminal investigation unit with an 'overcautious' attitude and therefore is extremely reluctant to exchange data.

p_1: Give me all data about trading in explosive materials.

When searching for data based on the request, agent o notices that his found data about trading in explosive materials has a risk factor since it can endanger the continuity of a local investigation. Therefore agent o initially rejects the request.

o_2: I will not give you this data.

After the refusal, the requesting agent p asks for the reason of the refusal

p_3: Why don't you give me that data?

Since all agents need to cooperate, for the proper execution of the police task, agent o gives the reason for his rejection.

o_4: Because it is forbidden to exchange data about trading in explosive materials.

Agent p asks for a reason why it is forbidden to exchange the data.

p_5: Why is it forbidden to exchange that data?

Agent o gives as reason that the action to exchange the data is forbidden since exchanging the data could endanger the continuity of an investigation.

o_6: Because exchanging the data could endanger the continuity of an investigation.

Agent p agrees that generally the requested data cannot be exchanged. However, in this case the data must be exchanged because claiming it is a matter of national importance.

p_7: You may be right in general but in this case it not forbidden to exchange the data because this is an exceptional situation since it is a matter of national importance.

Responding agent o reasons that national importance is more significant than the protection of a local investigation and he is therefore willing to exchange the data under the condition that the data cannot be exchanged with other police officers.

o_8: Ok, I admit that in this case it is not forbidden to exchange the data, so I retract that it is forbidden to exchange the data; I will give you this data on the following condition: the given data can not be exchanged with other police officers.

Agent p agrees with the offer.

p_9: I agree with this condition.

The dialogue is completed and responding agent p sends agent o the requested intelligence data about trading in explosive materials.

As discussed in Example 2, the initial refusal is based on article 13a, paragraph 3 of the Wpolr but in cases such as national importance, the rules forbidding the exchange of data can be set aside.

2.6 Relations between goals, actions and consequences

As stated in the introduction of this chapter, with the exchange of data police organizations generally try to achieve two – sometimes conflicting – goals. On the one hand, they have the goal to execute their appointed police task by exchanging data with other criminal investigation units. On the other hand, they have the goal to protect local investigations and informants. In this section an analysis of relations between goals, actions and possible consequences on other goals is presented. First the specific goals for criminal investigation units are presented (2.6.1). Then the types of actions concerning exchanging data are presented (2.6.2). Finally, relations between goals, the actions and the possible consequences are analyzed (2.6.3).

2.6.1 Goals

From the analysis of the police domain (the literature study and interviews with police officers[2]), three specific goals involved in severe-crime investigations can be

distinguished. The three goals are (1) the execution of the appointed police task, (2) the protection of individual privacy, and (3) the protection of local investigations and informants. The goal to execute the appointed police task is formulated in the Dutch police registers act (Wpolr). In the context of data exchange between CIEs, it means that CIEs are allowed to gather general intelligence data on severe crime in order to properly fulfill their task. The goal to protect personal privacy is also expressed in the Wpolr. This is based on the thought that a citizen needs protection against invasion of his private life by the government. The goal to protect privacy is not explicitly considered during the exchange between CIEs. However, the goal to protect privacy is an implicit general goal of the legal framework and intertwines with the other goals. For example, the goal to protect local investigations and informants has as consequence that the privacy of informants is also protected. Furthermore, as described in subsection 2.3.3 informants will not cooperate if their privacy is not protected and if they do not cooperate, the execution of the appointed police task is impeded (Ruth and Schreuders 2000). As a result the protection of privacy is not used as a ground for refusal in data exchange within the police organization (Ruth and Schreuders 2000). The goal to protect local police investigations and informants is derived from the criminal intelligence practice. In the Dutch police practice, the protection of investigations and informants is considered as a single goal[13] and is therefore combined as one goal. To summarize, for severe-crime data exchange between CIEs the two goals are:

- the proper execution of the appointed police task, and
- the protection of local investigations and informants.

2.6.2 Actions

In order to promote the goals concerning the exchange of severe-crime data, different types of local actions can be executed. Based on interviews with police officers, four local action types can be distinguished. The first action type is to exchange non-protected data, the second action type is to exchange protected data, the third action type is exchange protected data under a condition, and the fourth action is to refuse the exchange of protected data. Note that non-protected data is data which can be freely exchanged with other police officers. During the interaction of CIE agents, sometimes a non-local action is required to be executed by the other CIE agent as a condition before data is allowed to be exchanged. In the Dutch police domain, the condition is almost always that the other party is required to use the intelligence data only for analysis purposes. To summarize, the four local action types are:

[13] This is also described in the internal rapport 'Verstrekkingsafwegingen van Criminele Inlichtingen Eenheden Tussenrapportage Empirsch Onderzoek' written by Kielman and Koelewijn used for the ANITA research project meeting in Leiden March 13 2006.

- exchange non-protected data,
- exchange protected data,
- exchange protected data under analysis purposes condition, and
- refuse to exchange protected data.

2.6.3 Possible consequences on other goals

Next, an analysis of the relations between actions, goals, and consequences on other goals is presented. The local action types described in subsection 2.6.2 can have possible undesired consequences on other goals. A way to describe this relation is to list for each action type the possible consequences on all goals. The relation between actions and goals is domain dependent and therefore has to be based on empirical research. The goal shared by all CIEs is to properly execute the police task, which is promoted by exchanging data. From my analysis of the Dutch police domain, the relationships can be specified as follows:.

1. The action to exchange non-protected data has the likely consequences that it
    a. promotes the goal to properly execute the police task
    b. does not violate the goal to protect local investigations and informants
2. The action to exchange protected data has the likely consequences that it
    a. promotes the goal to properly execute the police task
    b. violates the goal to protect local investigations and informants
3. The action to exchange protected data under analysis purposes condition has the likely consequences that it
    a. promotes the goal to properly execute the police task
    b. does not violate the goal to protect local investigations and informants
4. The action to refuse to exchange protected data has the likely consequences that it
    a. violates the goal to properly execute the police task
    b. promotes the goal to protect local investigations and informants

When data is non-protected, the choice for exchanging that data is obvious since exchanging non-protected data both satisfies the goal to execute the police task and does not violate the goal to protect local investigations and informants. Refusing or stating that the non-protected data can be used for analysis purposes only is not sensible since exchanging that data does not violate the protection of local investigations. Therefore, for exchanging non-protected data the actions to refuse

non-protected data or to state that the non-protected data can be used for analysis purposes only is omitted in the specification.

Points 2, 3 and 4 specify the possible consequences of actions when exchanging protected data. The exchange of data can become problematic if the data is protected as a result of informant or local investigation protection. When the data is protected, the possible consequence of the action to exchange the data is the satisfaction of the goal to properly execute the police task; however the same action can have the consequence that it violates the goal to protect local investigations and informants. In most cases a request to exchange protected data will be (initially) refused since CIEs regard the protection of local informants as their primary goal (Kielman 2010).

The chosen action is based on which likely consequence on the goals is preferred. Preference in turn is based on the local rules and the local attitude of a criminal investigation unit. The attitude of CIEs is reflected in the local application of usage codes to data and the willingness to exchange protected data. For example, an overcautious criminal investigation unit will label data as protected more often than required. Table 1 shows the preferred action based on the role a CIE officer has (i.e. , requesting or responding agent). A CIE officer requesting data is not concerned with the protection of informants and investigations of the responding agent. Therefore, his preference is to always to exchange data and never refuse the exchange of data. The only case in which the requesting agent individually has to decide whether data can be exchanged, is when the data can be exchanged under a condition. This condition is an action the requesting agent has to perform, and his decision will be based on his local attitude. A responding CIE officer has a preference to exchange non-protected data. In all other cases, a responding CIE officer has to decide for each individual case whether to exchange or refuse to exchange data. Similar to a requesting CIE officer, this decision is based on his local rules and his attitude. For instance, a cooperative CIE officer is more easily persuaded to exchange protected data.

Table 1: preferred action based on role

	exchange non-protected data	exchange protected data	exchange protected data under condition	refuse to exchange protected data
requesting CIE agent	yes	yes	?	no
responding CIE agent	yes	?	?	?

Table 1 will be used for designing policies in Chapter 5, where every question mark in the table signifies when an agent individually has to decide whether data can be exchanged or can be exchanged under a condition.

2.7 Chapter summary

This chapter presented a description and analysis of regulated data exchange in the Dutch police domain. The chapter started with a discussion of the problem of regulated data exchange and continued with a description with the used information systems. Then the applicable Dutch law was presented and rewritten into an intermediate representation, which will be formalized in Appendix 1. Subsequently examples of dialogues were presented to illustrate the problem of regulated data exchange. The scenarios and the formalized rules will also be used to illustrate the communication protocol in Chapter 4 and the policies of Chapter 5. Finally, patterns in the examples (in terms of goals, actions, and possible consequences on other goals) were analyzed and these patterns will be used for designing policies in Chapter 5. First, however, a multi-agent system as a model for the regulated data exchange will be introduced in the next chapter.

Chapter 3

A multi-agent system for regulated data exchange

This chapter introduces a multi-agent system as a model for the exchange of severe-crime intelligence data in the Dutch police domain[14]. Section 3.1 discusses the most common definitions of agents and multi-agent systems. Section 3.2 describes how a multi-agent system can model regulated data exchange between organizations and argues why the Dutch police organization is a representative problem domain. Section 3.3 describes the requirements for a multi-agent system architecture, which can be implemented as a proof of concept system (see Chapter 6).

3.1 Agents and multi-agent systems

Multi-agent systems (MAS) provide a natural way to model distributed organizations in which people have different roles and goals. Although there is no single agreed definition of what an agent is, a commonly used definition originates from Wooldridge and Jennings (Wooldridge and Jennings 1995, page 4) who specify an agent as a hardware and/or software-based computer system which has the following properties:

- *"autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state (Castelfranchi 1995);*
- *social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language (Genesereth and Ketchpel 1994);*

[14] Note that this chapter is based on earlier work published in Dijkstra et al. (2005; 2006).

- *reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;*
- *pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative."*

Although different definitions and attributes are used to describe agents, the general consensus is that autonomy is central to the notion of agency (Weiss 1999) and that in almost all cases autonomous agents need to interact to fulfill their goals or improve their performance (Rahwan et al. 2004). Autonomy is a key characteristic that distinguishes agents from objects used in object-oriented software. While both objects and agents are defined as entities that encapsulate some state, are able to perform actions, and communicate by passing messages (Weiss 1999); it is the autonomy of agents that distinguishes them from objects, i.e., their ability to choose their actions. Agents send requests to other agents and the decision whether to perform the action is made by the agent receiving the request. By contrast, objects can invoke any publicly accessible method on any other object and the 'decision' is made by the external object invoking the method. Agents range from plain reactive agents which act in straightforward stimulus-response manner to agents which have additional human-like attributes, such as knowledge, beliefs and obligations (Shoham 1993). Other human-like attributes for agents are having a model of the environment, goals, plans to achieve those goals and being able to reason about actions. The selection of attributes for agents is made by the designer based on the relevant attributes in the chosen application domain. For example, for some applications the ability to learn from experiences is mandatory while for other applications it is undesirable (Carabelea and Boissier 2005).

Just as there is no single agreed definition of what a single agent is, there is no single definition of a multi-agent system. In the present research the definition (Wooldridge 2002) is adopted, who defines a multi-agent system as: 'a system that consists of a number of agents, which interact with one-another. In the most general case, agents will be acting on behalf of users with different goals and motivations. To successfully interact, they will require the ability to cooperate, coordinate, and negotiate with each other, much as people do'. Furthermore, Sycara identifies the following properties for a multi-agent system (Sycara 1998, page 80):

- *"each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint;*
- *there is no system global control;*

- *data is decentralized; and*
- *computation is asynchronous."*

In the next section, the elements of the definitions above will be applied to the Dutch police domain to illustrate the suitability of this domain.

3.2 Suitability of a MAS for regulated data exchange

The previous chapter described the problem of regulated data exchange between criminal investigation units. In essence, this problem is that only a part of what can be exchanged legitimately is actually exchanged (exchanges which are permitted do not occur in practice) and that sometimes data is exchanged illegitimately (exchanges which are forbidden do occur in practice). This problem is a consequence of the different goals that criminal investigation units try to achieve, even though criminal investigation units belong to the same organizations and they all need to exchange data for the goal to execute their appointed police task. They also have the goal to protect local investigations and informants, which can have the consequence that data is not exchanged which could be exchanged legitimately. Furthermore, sometimes data is exchanged illegitimately as a result of CIE officers not knowing all applicable regulations and the local interpretations of those regulations.

This section describes why a multi-agent system is suitable to model regulated data exchange in the Dutch police domain. To illustrate this, the elements of the definitions of agents and multi-agent systems are mapped to elements from the problem domain.

Data is decentralized

In the Dutch police domain, data is decentralized since criminal investigation units try to keep their data local in order to protect local investigations and informants. A way to decrease illegitimate data exchanges and increase legitimate data exchanges between criminal investigation units is to centralize all data. By centralizing data, all data is available to every criminal investigation unit and more data can be exchanged. The problem of more illegitimate data exchanges could be solved by assigning privacy officers to supervise the exchange of data. However, in practice this solution is not feasible. In the past, attempts have been made in the Dutch police domain to centralize data. These attempts caused criminal investigation units to keep their protected data local. The wish to control protected data by storing it locally is a direct result of the goal to protect their informants and local investigations. This shows that even if the criminal intelligence data were

physically centralized, criminal investigation units still wish to keep the data functionally decentralized so that the exchange of data is controlled by them..

Agents acting on behalf of users with different goals and motivations

While all CIE officers have the same two goals: the proper execution of the appointed police task by exchanging data and the protection of local investigations and informants, their preference is based on their role they play (see section 2.6 on page 37). On the one hand, a CIE officer in a requesting role is not concerned with the protection of informants and investigations of the responding CIE officer. Therefore, his goal is the proper execution of the appointed police task by always exchanging and never refusing the exchange of data with other criminal investigation units. On the other hand, a CIE officer in a responding role has to balance the two goals. Therefore software agents acting on behalf of their CIE officers have possibly conflicting goals as result of the different roles they play.

Autonomy; no system global control; pro-activity

In the Dutch police domain, criminal investigation units want to keep their autonomy from the other departments (i.e., they want to decide autonomously whether to exchange data, without the supervision of a centralized coordinator). Furthermore, CIE officers pro-actively search for severe-crime intelligence data from other criminal investigation units. The autonomy of CIEs is also related to the local interpretations of the applicable regulations. The legislator takes the interests of the criminal investigation units into account by formulating legal norms and central policies; these give room for local autonomy for fine tuning in local policies and individual decisions.

Each agent has incomplete information or capabilities for solving the problem

Each criminal investigation unit only has direct access to locally kept intelligence data and CIE officers do not always know the applicable regulations and the local interpretations of those regulations. Thus, CIE officers (and their supporting software agents) sometimes have incomplete information to be capable to help solving crime cases. In order to obtain all the available intelligence data, CIE officers need data held by other departments.

Interaction; social ability; reactivity

Since data is decentralized and CIE officers can have incomplete information to help solving crime cases, CIE officers need the social ability to interact in order to exchange data. CIE officers interact to coordinate their efforts to gather intelligence on severe crime. In most cases, CIE officers can request intelligence data and their requests are granted directly. In other cases, CIE officers negotiate and try to

persuade each other to determine if data can be exchanged. For instance, in Example 3 on page 36, CIE officers negotiate whether data about trading in explosive material can be exchanged. Initially, the CIE officer who is requested about exchanging data refuses. However, when the requesting CIE officer persuades him with the argument that the data about trading explosive materials is of national importance, he agrees to exchange his intelligence data. The interactions with negotiation and persuasion can be modeled in different ways, this will be further described in section 3.3 and those interactions are typical for what multi-agent systems try to model.

Computation is asynchronous

Each criminal investigation unit acquires intelligence data, and independently tries to help solve crime cases based on their own local data (i.e., they process their intelligence data asynchronously). However, severe crime is not always bound by the regions' criminal investigation units, therefore CIE officers need to coordinate their efforts in order to help solving crime cases (see section 2.1 on page 11). The coordination of CIE officers by engaging in dialogues is synchronous, since they have to wait for the responses of the other CIE officers before continuing.

As illustrated above, there is a strong relation between the attributes of multi-agent systems and the required functionalities for supporting regulated data exchange. A multi-agent system therefore provides a natural means to model the problem of regulated data exchange in the Dutch police domain. The CIE officers following their own goals and interacting with other CIE officers can be mapped directly on autonomous software agents. Agents can support regulated data exchange by autonomously engaging in dialogues on behalf of their CIE officers. Furthermore, the application of a multi-agent system makes it possible that the current distributed organizational and information structure of the CIEs is maintained. Thus, the knowledge of the local interpretations of the applicable regulations and intelligence data and are kept local and the decisions whether to exchange data is still made at the level of the criminal investigation units. Therefore, the Dutch police domain is quite suitable to determine whether a multi-agent system with agents who engage in dialogues can support regulated data exchange.

3.3 Requirements for the MAS architecture

In this section the requirements for the multi-agent system architecture in order to be a tenable model of regulated data exchange in police practice are presented. Briefly, based on the examples of section 2.5 on page 33, the requirements are that the agents (A) have knowledge, (B) have goals, (C) are able to reason, and (D) are able to interact.

A: Knowledge

For agents to be able to regulate distributed data exchange, the agents must have knowledge of the relevant regulations and the local interpretations of those regulations, their goals and the likely consequences of their actions. Below the knowledge requirements are further described.

Knowledge of the relevant regulations and the local interpretations

All agents need to have knowledge of the relevant regulations and each agent must have knowledge of the local interpretations of those regulations. Furthermore, each agent must have knowledge of the local intelligence data. Data will only be exchanged if the local agent decides to do this. Finally, in order to represent knowledge and being able to exchange data, the agents must have an ontology (Gruber 1993) of the domain.

B: Goals

The agents in the problem domain have - possibly conflicting - goals. As described in section 3.2, there are two agent roles: a requesting agent who has the goal to properly execute his appointed police task by exchanging intelligence data with other criminal investigation units and a responding agent who has to balance out the goal to execute his appointed task by exchanging intelligence data and the goal to protect his local investigations and informants. The agents and their interactions should be designed in such a way that their behavior agrees with their goals.

The likely consequences of their actions

Agents representing CIE officers should have knowledge of the possible consequences of their actions for the realization of their goals (see section 2.6 on page 37 for an analysis). A typical example of the likely consequences of actions is the satisfaction of the goal to execute the appointed police task by exchanging data. However, the same action can have the consequence that it conflicts with the goal to protect local investigations and informants.

C: Reasoning

Agents must be able to reason with their knowledge of the relevant regulations, their goals they try to realize and the likely consequences of their actions. As illustrated by the examples in section 2.5, the interaction between agents often involves argumentation. For instance, in Example 3 (page 36), CIE officers negotiate and try to persuade each other whether intelligence data about trading in explosive material can be exchanged. Therefore, the agents should be capable of generating and evaluating arguments for and against certain claims and they must be able to revise their beliefs as a result of the dialogues. The agents especially

reason about whether something is obligatory, forbidden, or whether something violates their interest. With respect to the latter, it is useful to remark that conflicts between arguments on this issue can be resolved in the reasoning process.

D: Interaction

As described in the examples, agents engage in dialogues to negotiate and persuade each other whether intelligence data must be exchanged. Therefore, the interactions are modeled as dialogues between agents. A well known typology of dialogue types is defined by Walton and Krabbe (1995). They distinguish six dialogue types depending on the goal of the dialogue. The goal of information-seeking dialogues is exchanging information, the goal of negotiation dialogues is resolving a conflict of interests, the goal persuasion dialogues is resolving differences of opinion, the goal of deliberation dialogues is making a decision on a course of action, the goal of inquiry dialogues is the growth of knowledge and the goal of eristic dialogues is to vent grievances.

During a dialogue, the responding agent's goals sometimes lead him to state conditions under which he is willing to exchange data. Therefore, the agents must be able to negotiate with each other (see Example 1 on page 33). In addition, the responding agent may be mistaken in believing that he must or should not exchange the requested data (see Example 2 on page 34). So, the agents must be able to engage in persuasion dialogues. Thus, three types of dialogues are necessary to model the regulated data exchange: (1) information-seeking, (2) negotiation, and (3) persuasion dialogues.

There are (at least) two ways to model the relation between the three types of (information-seeking, negotiation, and persuasion) dialogues, depending on two ways to model the start of a dialogue in this domain.

i. The first interpretation is that each dialogue starts as an information-seeking dialogue. It then shifts to another type of dialogue, either persuasion or negotiation. It shifts to a persuasion dialogue if the responding agent states he will not grant the request since doing so would have negative consequences for his investigations and the requesting agent starts to persuade the responding agent that he is wrong about this. The dialogue instead shifts to negotiation if the requesting agent promises to do or refrain from doing something on the condition that the responding agent gives him the data. Further shifts may occur, for instance, from a persuasion to a negotiation dialogue or vice versa.

ii. The second interpretation is that each dialogue starts as a negotiation, viz. as a request to exchange data about something. Such a dialogue may shift to embedded persuasion if the requesting agent tries to persuade the responding agent that he is

wrong about a rejection, e.g., that granting would endanger the protection of local investigations. After the persuasion terminates the interrupted negotiation resumes. If that terminates successfully, a (trivial) information-seeking dialogue starts; its termination also terminates the overall interaction. In the literature, interactions of type (ii) are also referred to as 'service request dialogues' (Bentahar et al. 2007; Torroni et al. 2007).

In the police domain, most dialogues are of type (ii) since usually the requesting agent will not only ask a question but will inquire whether the other agent is willing to provide him with a certain body of data. This seems more like negotiation than like information-seeking. Therefore, in this research the focus on will be on dialogues of the second type. To enable such interactions where reasons for rejections are given which can become the subject of the dialogue, a suitable dialogue protocol must be implemented. In addition, the agents must be given policies for their behavior in the dialogues. These policies should be designed to further the agent's goals. Since these goals also include the goal of the overall organization (i.e., the execution of the appointed police task), the agents' policies should induce a fair degree of cooperativeness.

3.4 Chapter summary and preview

This chapter described how a multi-agent system can model regulated data exchange between organizations and argued why the Dutch police organization is a representative problem domain. The examples of dialogues about regulated data exchange from chapter 2 were used to clarify the functional requirements. The examples illustrate that the agent interactions often involve argumentation about regulated data exchange, and that they can be interpreted as negotiation with embedded persuasion dialogues. The four requirements of multi-agent system (knowledge, goals, reasoning, and interaction) will be formalized in the next two chapters. Chapter 4 formalizes the requirements concerning knowledge, goals, the likely consequences of actions and the ability to reason and engage in dialogues. Chapter 5 formalizes the requirement that agents and their interactions should be designed in such a way that their behavior agrees with their goals. For this negotiation and persuasion dialogue policies will be specified. The key idea of the dialogue policies is that they prescribe the best reaction during the dialogues, i.e., the reaction which is expected to further the different goals while respecting the attitude of the criminal investigation unit towards data exchange.

Chapter 4

Formalization of the multi-agent system

This chapter describes the formalization of the requirements for a multi-agent system supporting regulated data exchange with argumentation[15]. In section 4.1, the chosen logical argument system to model the internal inferences of an agent is presented, while in section 4.2, the dialogue system to model the interaction between agents to support regulated data exchange is specified.

4.1 Logics for defeasible argumentation

This section describes logics for defeasible argumentation. For overviews see Prakken and Vreeswijk (2002) and various chapters in Rahwan and Simari (2009). First ASPIC as a logic for defeasible argumentation and the representation of knowledge in ASPIC is discussed (4.1.1). Then the representation of deontic modalities and agent goals is specified (4.1.2).

Nonmonotonic (or defeasible) reasoning is a form of reasoning in which conclusions can be withdrawn when new information becomes available. A straightforward example of defeasible reasoning in the Dutch police domain is a CIE officer who deliberates whether he can exchange data labeled with usage code *11* (ready for operational usage, see chapter 2 on page 27). Normally he reasons that the data can be exchanged. However, when the data is no longer up to date, the CIE officer also concludes that the same data cannot be exchanged. Assuming that the reason for not exchanging data is stronger, the previous conclusion is withdrawn. A second example is the national importance scenario described in Example 3 on page 36. In this example, a responding CIE officer first reasons that he cannot exchange intelligence data about trading in explosive materials. However, when the requesting agent tells him that he needs the data for national importance, the responding CIE officer reasons that this is an exception to the general rule.

[15] Note that this chapter is based on earlier work published in (Dijkstra et al. 2005; Dijkstra et al. 2006; Dijkstra et al. 2007).

Therefore, in that specific case, the general rule is not applicable and he is willing to exchange the requested data. Logics for defeasible argumentation (or argumentation systems for short) conceptualize and formalize this type of reasoning as the construction and comparison of arguments for and against a certain conclusion. The nonmonotonicity is conceptualized in terms of interactions between conflicting arguments: that is, when new information allows for new stronger arguments that defeat the original argument. An argumentation system has a logical language to represent the knowledge, and arguments are defined in terms of the chosen logical language. An elementary argument has a conclusion, premises, and an inference connecting the conclusion. A complex argument can be constructed by chaining elementary arguments.

To illustrate how argumentation systems formalize defeasible reasoning, consider the following example. Assume that there is a collection of arguments that can be constructed, where some arguments attack and other arguments support each other. Furthermore, assume that in this collection, some arguments are for conclusion *c* and other arguments are against conclusion *c*. What can be concluded about conclusion *c* in this situation? Argumentation systems formalize this problem.

Arguments can attack each other in different ways. In the literature, three kinds of attack are distinguished: when arguments have contradictory conclusions *(rebut)*, when the conclusion of an argument denies a premise of another argument *(premise-attack)* and when an argument shows that an inference is incorrect *(undercut)*. Argumentation systems use a binary defeat relation to determine which argument of two attacking arguments wins. The criteria for the defeat relation are domain-specific. For example, a criterion can be that the most specific argument wins. A second example of a criterion is that an argument originating from someone with a higher authority wins.

Given a set of arguments and a binary defeat relation defined over it, an argumentation system classifies arguments into three types: winning *(justified),* losing *(overruled)* and ties *(defensible)*. This classification can be seen as the inference relation of this system. One way to define the inference relation of an argumentation system is in the dialectical form of an argument game. In an argument game, defeasible reasoning is modeled as a dispute between a proponent and opponent of a claim. The proponent defends the claim by constructing arguments and the opponent attacks the claim by constructing counterarguments. The claim is proven, if for some argument for the claim, the proponent has a winning strategy. The interpretation of proven is having a defensible or justified argument, which is dependent on what the argument game tries to formalize.

4.1.1 ASPIC as a logic for defeasible argumentation

In this section ASPIC as a logic for defeasible argumentation and the representation of knowledge is presented. First some background information of the ASPIC argumentation system is given (A), after which the representation of knowledge using the ASPIC logic is described (B). Then it is specified how arguments can be formed (C) and how they can defeat each other in ASPIC (D). Finally the supported argument games are discussed (E).

A: Background ASPIC

In order to illustrate how a multi-agent system described in chapter 3 can support regulated data exchange, a proof-of-concept implementation is needed. Furthermore, since the internal reasoning of CIE officers is modeled as defeasible argumentation, an argumentation system to support this type of reasoning is required. For the proof-of-concept implementation, the ASPIC argumentation system is used, which was developed within the ASPIC project (IST-FP6-002307). ASPIC is an inference engine for an argument-based nonmonotonic logic and is based upon the model described in Amgoud et al. (2006) and the algorithms specified in Vreeswijk (2006). The tool computes the dialectical status of arguments according to grounded or preferred semantics Dung (1995). Knowledge is represented in the prolog-like syntax of the ASPIC logic. As in prolog, function symbols, predicate symbols, and constants start with a lower case letter, variables start with a capital letter or underscore, and negation is expressed through the ~ character. Using a degree of belief, the strength of rules and facts can be evaluated. In ASPIC the degree of belief is expressed by a number greater than 0 and less or equal to 1, where 1 is the default value. In order to decide conflicts between rules, preference rules can be implemented using the degree of belief. ASPIC also supports rule naming with variables, which makes it possible to refer to rules in other rules. Variables in rule names allow for distinguishing between rule schemes and particular instances of rules. To enable talking about the validity of rules, antecedents of rules will be extended with an additional condition '`valid(r(X))`'.

B: Rules and factual statements

Let us now look how knowledge can be represented as rules and factual statements in the ASPIC logic. Two examples will be provided. In the first example, the domain rule stating that if the requested data has source protection then it is forbidden to exchange the data (see Appendix 1, page 166) can in ASPIC be represented as:

```
[rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT)]
obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
```

```
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT))
0.6.
```

The consequent of the rule is:

```
obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID))).
```

The antecedents of the rule are:

```
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)), and
valid(rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT)).
```

The reason for using the holds and obliged predicates will be explained in subsection 4.1.2. In this example, the rule is named:

```
rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT)
```

by preceding the rule with

```
[rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT)].
```

The degree of belief in this rule is '`0.6`', which is added to the end of the rule. The additional valid condition which encloses the name of the rule by the valid predicate is

```
valid(rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT))
```

A second example is the factual statement that data-unit identifier[16] `2` has source protection, which in ASPIC can be represented as:

```
holds(sourceProtection(2, adriana)).
```

The factual statement that agent `p` has requested agent `o` to exchange data-unit identifier `2` matching the query about Adriana, can in ASPIC be represented as:

```
requested(exchangeData(p, o, adriana, 2)).
```

For every rule with an additional `valid` condition, a corresponding additional factual statement can be added to match the valid condition, which in this case is:

[16] Data-unit identifiers are described in subsection 6.1.1.

```
valid(rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT))
0.6.
```

Note that only rules can be named in ASPIC, therefore facts are not preceded by rule names in the knowledge representation.

C: Arguments

In the context of inference, arguments provide reasons to believe in a conclusion. As described in section 4.1, an elementary argument has a conclusion, premises, and an inference connecting the conclusion. A complex argument can be constructed by chaining elementary arguments. Conceptually, in ASPIC two types of premises can be distinguished. Factual statements are premises which represent facts, and rules are premises which represent (domain) rules. However, technically, rules are domain-specific inference rules, just as, for example in default logic (Reiter 1980).

According to ASPIC an argument is defined as follows: for a given argument, the function PREM returns all the formulas of the knowledge base *KB* (called premises) used to build the argument, CONC returns its conclusion, and SUB returns all its subarguments. According to Amgoud et al. (2006, Definition 12), an argument can be defined as follows.

Definition 1 Arguments

An argument A constructed from KB (where KB consists of KB_R representing the rules and KB_F representing the facts) is defined as

P if $P \in KB_F$ with:

$\text{PREM}(A) = \{P\}$
$\text{CONC}(A) = P$
$\text{SUB}(A) = \{P\}$

P since $A_1 \ldots A_n$ if $A_1 \ldots A_n$ are arguments constructible from KB such that there exists a rule $P \leftarrow \text{CONC}(A_1), \ldots, \text{CONC}(A_n)$ in KB_R with

$\text{PREM}(A) = \text{PREM}(A_1) \cup \ldots \cup \text{PREM}(A_n)$
$\text{CONC}(A) = P$
$\text{SUB}(A) = \text{SUB}(A_1) \cup \ldots \cup \text{SUB}(A_n) \cup \{A\}$

For notational convenience, the inference rules will be listed together with the premises. For example, using the previous rule and factual statements, $argument_1$ can be constructed as follows:

```
obliged(not(exchangeData(p, o, adriana, 2)))
since

obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT)).

holds(sourceProtection(2, adriana)).
requested(exchangeData(p, o, adriana, 2)).
valid(rSourceProtection_F(p, o, adriana, 2, adriana)).
```

In $argument_1$, the conclusion is:

```
obliged(not(exchangeData(p, o, adriana, 2)))
```

The rule premise of $argument_1$ is:

```
obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT))
0.6.
```

The factual statements (fact premises) of $argument_1$ are:

```
holds(sourceProtection(2, adriana))., 
requested(exchangeData(p, o, adriana, 2)). and
valid(rSourceProtection_F(p, o, adriana, 2, adriana))..
```

The degree of belief of $argument_1$ is '`0.6`' since the degree of belief of an argument is equal to the weakest degree of belief of its constituting elements.

D: Conflict relations between arguments

Let us now specify the conflict relations between arguments. If $argument_a$ has an equally strong or stronger degree of belief than $argument_b$ then the following defeat relations between those arguments are possible: $argument_a$ successfully *rebuts* $argument_b$ if the arguments have contradicting conclusions. $Argument_a$ successfully *undercuts* $argument_b$ if $argument_a$ excludes a rule used in $argument_b$ by stating that the rule is not applicable; $argument_a$ successfully *premise-attacks* $argument_b$ if the

conclusion of $argument_a$ contradicts $premise_p$ of $argument_b$ and the degree of belief of $argument_a$ is stronger than $premise_p$ of $argument_b$. As discussed in the construction of $argument_1$ above, the degree of belief of an argument is equal to the weakest degree of belief of its constituting elements. Therefore, even when $argument_a$ has an equally strong or stronger degree of belief than $argument_b$, $premise_p$ of $argument_b$ still can have a higher degree of belief than $argument_a$. The defeat relations also have an indirect version, i.e., attacking a subargument of an argument. For example, if $argument_a$ defeats $subargument_b$ then $argument_a$ indirectly defeats $argument_b$. The defeat relation is the evaluation of the relative strength of a pair of conflicting arguments. However, the ultimate status of an argument is dependent on the interaction of all available arguments. For example, if $argument_b$ defeats $argument_a$ but $argument_b$ itself is defeated by $argument_c$ then $argument_c$ *reinstates* $argument_a$.

Rebuttal

Let us next look how an argument can be rebutted in the ASPIC logic. An argument ($argument_2$) will be constructed which rebuts $argument_1$. The domain rule stating that if the requested data originating from the informant have source protection and the CIE officer uses the data for analysis purposes only, then it is not forbidden to exchange data (see page 166) can in ASPIC be represented as:

```
[rSourceProtectionAnalysis_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER, INFORMANT)]
~obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
holds(analysisPurposesOnly(CIE_OFFICER, DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rSourceProtectionAnalysis_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER, INFORMANT)) 0.7.
```

The factual statements stating that data labeled with data-unit identifier 2 have source protection and that CIE officer Dwight uses data labeled with data-unit identifier 2 for analysis purposes only, can be represented in ASPIC as:

```
holds(sourceProtection(2, adriana)).
holds(analysisPurposesOnly(cie_officer_dwight, 2)).
```

Using the above rule and factual statements, $argument_2$ can be constructed:

```
~obliged(not(exchangeData(p, o, adriana, 2)))
since

~obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
```

```
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
holds(analysisPurposesOnly(CIE_OFFICER, DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rSourceProtectionAnalysis_P((S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER, INFORMANT)).

holds(sourceProtection(2, adriana)).
holds(analysisPurposesOnly(cie_officer_dwight, 2)).
valid(rSourceProtectionAnalysis_P(p, o, adriana, 2,
cie_officer_dwight, adriana)).
```

$Argument_2$ successfully rebuts $argument_1$ since the arguments have conflicting conclusions and the degree of belief in $argument_2$ is stronger than the degree of belief in $argument_1$. The degree of belief of $argument_2$ is '`0.7`', which originates from the degree of belief of the rule premise. The degree of belief of $argument_1$ is '`0.6`', which also originates from the degree of belief of the rule premise.

Undercut

Each defeasible rule can have one or more undercutters, which specify under which circumstances the inference rule is not applicable. To be able to argue about the applicability of an inference rule, the rule first has to be established as legally valid. An inference rule is legally valid if its source is legally recognized (such as the applicable law or being stated by a legal authority). It is also possible to argue about the legal validity of domain rules, see the 'giving grounds for a rule' section on page 60. In the next example, however, the validity of the domain rules is assumed by accompanying them with a factual statement matching the valid condition. The example illustrates how an argument can be undercut in the ASPIC logic. An argument ($argument_3$) will be constructed which undercuts $argument_2$. Assuming the domain rule stating that if a CIE officer is under internal investigation, the rule stating: 'if data has source protection and the CIE officer uses the data for analysis purposes only, then it is not forbidden to exchange the data' cannot be applied, which can be represented in ASPIC as follows:

```
[r3a(S, R, QUERY, DATA_UNIT_ID, CIE_OFFICER, INFORMANT)]
~rSourceProtectionAnalysis_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER, INFORMANT)
<-
holds(internalInvestigation(CIE_OFFICER)),
valid(r3a(S, R, QUERY, DATA_UNIT_ID, CIE_OFFICER, INFORMANT)) 0.8.
```

The domain rule stating 'if an internal investigation has been started for a CIE officer, then the officer is under internal investigation' can be represented as:

```
[r3b(CIE_OFFICER)]
holds(internalInvestigation(CIE_OFFICER))
```

```
<-
holds(startedInternalInvestigation(CIE_OFFICER)),
valid(r3b(CIE_OFFICER)) 0.85.
```

The factual statement that an internal investigation has been started for CIE officer Dwight can be represented in ASPIC as:

```
holds(startedInternalInvestigation(cie_officer_dwight)).
```

The factual statements that rules r_{3a} and r_{3b} are valid can be represented as:

```
valid(r3a(o, p, adriana, 2, cie_officer_dwight)).
valid(r3b(cie_officer_dwight)).
```

Using the above rules and factual statements, $argument_3$ can be constructed:

```
~rSourceProtectionAnalysis_P(p, o, adriana, 2, cie_officer_dwight)
since

~rSourceProtectionAnalysis_P(rSourceProtectionAnalysis_P(S, R,
QUERY, DATA_UNIT_ID, CIE_OFFICER)
<-
holds(internalInvestigation(CIE_OFFICER)),
valid(r3a(rSourceProtectionAnalysis_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER)).

holds(internalInvestigation(CIE_OFFICER))
<-
holds(startedInternalInvestigation(CIE_OFFICER)),
valid(r3b(CIE_OFFICER)).

valid(r3a(o, p, adriana, 2, cie_officer_dwight)).
valid(r3b(cie_officer_dwight)).
holds(startedInternalInvestigation(cie_officer_dwight)).
```

Based on the validity of the *rSourceProtectionAnalysis_P* domain rule (which is used to construct $argument_2$) and the validity of rules r_{3a} and r_{3b} (which are used to construct $argument_3$), both arguments are valid. Although the *rSourceProtectionAnalysis_P* domain rule is valid, there can be circumstances where this rule is not applicable. Those circumstances are specified by the instantiated rules r_{3a} and r_{3b}. Therefore, at this point $argument_3$ successfully undercuts $argument_2$, which in turn reinstates $argument_1$.

Premise-attack

Let us look how an argument can premise-attack another argument in the ASPIC logic. The domain rule stating that if a CIE officer is cleared from investigation, then the CIE officer is no longer under internal investigation and the factual

statement that CIE officer Dwight is cleared from internal investigation are represented in ASPIC as:

```
[r4(CIE_OFFICER)]
~holds(internalInvestigation(CIE_OFFICER))
<-
holds(clearedInternalInvestigation(CIE_OFFICER)),
valid(r4(CIE_OFFICER))0.9.

holds(clearedInternalInvestigation(cie_officer_dwight)).
```

Now $argument_4$ can be constructed:

```
~holds(internalInvestigation(cie_officer_dwight))
since

~holds(internalInvestigation(CIE_OFFICER))
<-
holds(clearedInternalInvestigation(CIE_OFFICER)),
valid(r4(CIE_OFFICER)).

holds(clearedInternalInvestigation(cie_officer_dwight)).
valid(r4(cie_officer_dwight)).
```

At this point $argument_4$ successfully premise-attacks $argument_3$, given that the conclusion of $argument_4$ conflicts with a premise of $argument_3$ and the degree of belief in $argument_4$ is not lower than the conflicting premise of $argument_3$. Therefore, $argument_2$ is reinstated.

Giving grounds for a rule

Since the knowledge in the police domain is not only elicited from the applicable law but also from (possibly contradictory) local interpretations, it should be possible to argue about the validity of those different local interpretations. In the previous formalized examples, every rule is accompanied with a factual statement matching the additional valid condition. To enable arguing about grounds for a rule, the factual statements about the validity can be replaced by rules. For example, in rule r_4 the matching factual statement is '`valid(r4(CIE_OFFICER))`'. To enable arguing about the validity of rule r_4, the matching factual statement is replaced by rule r_5 where its conclusion is a ground for r_4. An example of a possible ground for rule r_4 is that the decision for clearance has been granted by an internal investigations organization (denoted by the `ORG` variable). This example can be formalized in ASPIC as:

```
[r5(CIE_OFFICER, ORG))]
valid(r4(CIE_OFFICER))
<-
holds(decisionGranted(CIE_OFFICER, ORG)),
```

```
valid(r5(CIE_OFFICER, ORG))0.9.

holds(decisionGranted(cie_officer_dwight, org)).
valid(r5(cie_officer_dwight, org)).
```

Of course, also further grounds for r_5 can be given. From the previous rules and factual statements argument$_5$ can be constructed:

```
valid(r4(cie_officer_dwight))
since

valid(r4(CIE_OFFICER))
<-
holds(decisionGranted(CIE_OFFICER, ORG)),
valid(r5(CIE_OFFICER, ORG)).

holds(decisionGranted(cie_officer_dwight, org)).
valid(r5(cie_officer_dwight, org)).
```

E: Argument games

In the final stage of argumentation, argument games are used to determine the dialectical status of an individual argument. Given a collection of arguments, the defeat relations and the semantics it is now possible to define the acceptability of an individual argument. The ASPIC inference engine supports grounded (skeptical) and preferred credulous semantics; see section 2.5 of ASPIC deliverable D 2.1 (Amgoud et al. 2004) for a specification of the algorithms. The first argument game supported by ASPIC is the argument game with grounded semantics, where only justified arguments are accepted, i.e., whether it is a member of the (unique) grounded extension. An argument is overruled if it is defeated by a justified argument and not in the grounded extension and an argument is defensible otherwise. The second supported argument game is with preferred semantics, where also defensible arguments are accepted. The argument game with grounded semantics is more restrictive. In this chapter, always one of the conflicting example arguments is stronger, in those examples the chosen semantics do not give rise to different outcomes. However, if the examples are equally strong and therefore mutually defeating, then with grounded semantics both arguments are not acceptable while with preferred credulous semantics both arguments are acceptable.

4.1.2 Representation of deontic modalities and agent goals

This subsection discusses the choices for the representation of deontic modalities and the goals of agents in the ASPIC logic. The goals are represented in the agent's internal knowledge base, together with knowledge that is relevant to respecting the agent's goals, such as knowledge on when exchanging data is obligatory, permitted or forbidden, and the effect of exchanging data on the other goal of the agent.

Deontic modalities

In standard deontic logic, the deontic notion *obliged* can be used to define the other two notions *forbidden* and *permitted* (McNamara and Prakken 1998). Using *obliged p*, *forbidden p* is then defined as *obliged* ¬*p*, and *permitted p* in turn is defined as ¬*forbidden p*:

Obliged	$O(p)$
Forbidden	$F(p) = O(\neg p)$
Permitted	$P(p) = \neg F(p)$

The deontic modalities *obliged*, *forbidden*, and *permitted* are not part of the ASPIC logic. However, by using reification the meaning of deontic operators can be respected. Reification enables reasoning about statements. Formulas from the object language are converted into terms, which can be used as terms in the scope of first-order predicates. With reification, the deontic notions from standard deontic logic are defined as first-order predicates in the ASPIC logic. In the knowledge representation, only the *obliged* operator is used. Therefore, in every rule '`forbidden(P)`' is substituted with '`obliged(not(P))`' and '`permitted(P)`' is substituted with '`~obliged(not(P))`'. The *holds* predicate is used to simulate modalities, since distinguishing that *p* is true and that *p* is obligatory (or that *p* is a goal) requires that *p* has to be a term in both formulas. With the addition of the holds predicate, the expressions that *p* is true and *p* is obligatory (or a goal) then can stated as '`holds(P)`' and '`obliged(P)`' (or '`goal(P)`') in the ASPIC logic. To preserve the meaning of *holds, obliged*, and *not*, the following axioms are necessary:

```
~holds(not(P)) <- holds(P).
~holds(P) <- holds(not(P)).

~obliged(not(P)) <- obliged(P).
~obliged(P) <- obliged(not(P)).
```

Let us look at an example of a domain rule to illustrate how to reify deontic modalities in the ASPIC logic. The domain rule stating that if data has usage code *11*, then it is obliged to exchange the data (see page 162) can be represented with reification as:

```
[rDataUsageCode11_O(S, R, QUERY,DATA_UNIT_ID)]
obliged(exchangeData(S, R, QUERY, DATA_UNIT_ID))
<-
holds(dataUsageCode11(DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rDataUsageCode11_O(S, R, QUERY, DATA_UNIT_ID)) 0.5.
```

In the example, the variable *S* denotes the responding agent and the variable *R* denotes the requesting agent. Furthermore, the expression '`exchangeData(S, R, QUERY, DATA_UNIT_ID)`' is regarded as a term from the object language, which makes it possible to determine which deontic predicate is applicable, which in this case is '`obliged`'.

Agent goals

Let us now illustrate how goals and violations of those goals can be represented in the ASPIC logic. Note that other agent research has been done on agent goals, for instance Belief-Desire-Intention agents (see for example Rao and Georgeff 1995). In the present research, for practical purposes goals are simply expressed with reification. An example of an agent goal is the protection of his local investigations and informants (see page 171), which can be represented in ASPIC as:

```
[rProtectionGoal(S, DATA_UNIT_ID, INFORMANT)]
goal(S, protection(DATA_UNIT_ID, INFORMANT))
<-
holds(informantOf(INFORMANT, DATA_UNIT_ID, S)),
valid(rProtectionGoal(S, DATA_UNIT_ID, INFORMANT)).
```

The domain rule *rProtectionGoal* states that if data originates from an agent's informant, then the goal of the agent is to protect the informant. Let us now see how an agent is able to infer that his interests are violated. The violation of interests rule states that if something is (not) a goal of an agent but the opposite holds, then his interests are violated, which can be represented as (see page 172):

```
[rViolationOfOwnInterestsA(S, P)]
violationOfOwnInterests(S)
<-
holds(not(P)), goal(S, P).

[rViolationOfOwnInterestsB(S, P)]
violationOfOwnInterests(S)
<-
holds(P), goal(S, not(P)).
```

To illustrate how an agent goal relates to a violation of an agent's interests, consider the following example. The domain rule stating that if data has source protection and the data is exchanged then the informant who has supplied the data has no protection (see page 165) can be represented as:

```
[rSourceProtection_NP(S, R, QUERY, DATA_UNIT_ID, INFORMANT)]
holds(not(protection(DATA_UNIT_ID, INFORMANT)))
<-
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
happens(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
```

```
valid(rSourceProtection_NP(S, R, QUERY, DATA_UNIT_ID, INFORMANT)).
```

When rule *rSourceProtection_NP* is applicable there is a violation of interests since the goal of the agent to protect informants is violated by the fact that the informant has no protection.

In chapter 5, goals, violations of interests and deontic modalities are used in the deliberation process of agents to determine whether intelligence data can be exchanged. In the next section, the dialogue system for argumentation is discussed that will enable dialogues between software agents.

4.2 Dialogue system for argumentation

This section presents the dialogue system for the agent interaction. Subsection 4.2.1 specifies the communication language and subsection 4.2.2 describes the protocol. This section builds on the work of Van Veenen and Prakken (2006) and Prakken (2005). The dialogue system is adapted to be compatible with the ASPIC inference engine and to support the regulated data exchange between CIE officers.

Whereas an argument game system models the internal inferences of an agent as a dialectical argument game, a dialogue game system models the dialogical interaction as dialogue game between agents. Dialogue systems have a *topic language* L_t with an associated *logic* $\boldsymbol{L}$ to represent and reason with domain knowledge and a *communication language* L_c to specify the speech acts, i.e., the performatives and their propositional content represented in the topic language. Furthermore, a communication language has an associated *protocol P* to specify which moves are allowed at each point in a dialogue. Dialogue systems also have *effect rules* to specify the effects of utterances on the players' commitments, and *outcome rules* to define the outcome of a dialogue.

The dialogue system of Van Veenen and Prakken (2006) combines a negotiation protocol of (Wooldridge and Parsons 2000) with a persuasion language and protocol of (Prakken 2005). The system of Van Veenen and Prakken (2006) seems appropriate for the modeling purposes since, as described in section 3.3, the agent interactions typically have the form of negotiation with embedded persuasion. Also, this embedding occurs especially when the requesting agent asks why the responding agent rejects the request for data, and this is precisely the kind of embedding modeled in Van Veenen and Prakken (2006). Since the dialogue systems are defined within Prakken's framework (Prakken 2005), let us now look at the formal definitions of the framework

As for the logic for defeasible argumentation, Prakken uses a general definition for the logic of defeasible argumentation. In the present research, ASPIC is used as the argumentation system.

Definition 2 Communication language

A *communication language* is a set of L_c *of speech acts*. Each speech act $s \in L_c$ is of the form *p(content)* where p is an element of a given set P of performatives and *content* is defined in topic language L_t with associated logic $\boldsymbol{L}$. On L_c a reply structure is assumed, where each move is either an attack or a surrender to its target. The protocol for L_c is defined in terms of the notion of a *dialogue*, which in turn is defined in terms of a *move*.

Definition 3 Moves and dialogues

The set M of moves is defined as $N \times \{P, O\} \times L_c \times N$, where the four elements of a move are denoted by:

- $id(m)$, the *identifier* of the move,
- $pl(m)$, the *player* of the move,
- $s(m)$, the *speech act* performed in the move,
- $t(m)$, the *target* of the move.

The set of *dialogues*, denoted by M^*, is the set of all sequences $m_1, ..., m_i, ...$ from M such that:

- each i^{th} element in the sequence has identifier i,
- $t(m_1) = 0$,
- for all $i > 1$ it holds that $t(m_i) = j$ for some m_j preceding m_i in the sequence.

For any dialogue $d = m_1, ..., m_n, ...$ the sequence $m_1, ..., m_i$ is denoted by d_i, where d_0 denotes the empty dialogue. When $t(m) = id(m')$ then m *replies* to m' in d and that m' is the target of m in d. Sometimes the notation is slightly abused to let $t(m)$ denote a move instead of just its identifier. When $s(m)$ is an attacking (surrendering) reply to $s(m')$ then m is an attacking (surrendering) reply to m'. A protocol also assumes a turntaking rule to assign a player for every move.

A protocol is a function *Pr* that for each legal finite dialogue specifies the legal next moves. If it returns an empty set, then and only then, the dialogue is terminated. For a complete definition of protocols, see Definition 7 in Prakken (2005). All protocols are further assumed to satisfy the following basic conditions for all moves m_i and all legal finite dialogues d.

If $m \in Pr(d)$, then:

- R_1: $pl(m) \in T(d)$;
- R_2: If $d \neq d_0$ and $m \neq m_1$, then $s(m)$ is a reply to $s(t(m))$ according to L_c;
- R_3: If m replies to m', then $pl(m) \neq pl(m')$;
- R_4: If there is an m' in d such that $t(m) = t(m')$ then $s(m) \neq s(m')$;
- R_5: For any $m' \in d$ that surrenders to $t(m)$, m is not an attacking counterpart of m'.

Rule R_1 states that a move is legal only if moved by the player-to-move (T returns the player whose turn it is to move in dialogue d). R_2 says that a replying move must be a reply to its target according to communication language L_c, and R_3 states that a player cannot reply to one's own moves. Rule R_4 states that if a player backtracks, the new move must be different from the first one. (in Prakken's framework backtracking is interpreted as any alternative reply to the same target in a later turn). Finally, R_5 says that once a move is surrendered, it may not be 'revoked'. Below the communication languages (4.2.1) and the communication protocols (4.2.2) are discussed.

4.2.1 Communication languages

In this section the sublanguages for negotiation and persuasion and their combination are defined.

As specified in Definition 2, a communication language consists of *speech acts p(content)* where p is a performative and *content* is defined in topic language L_t with associated logic $\boldsymbol{L}$. In the persuasion communication language, *content* has one element *Arg* or *P* to denote an argument or proposition. For the negotiation communication language, the elements of *content* are literals *A* and *C* which denote the action and the condition (except for *withdraw* speech acts). (Literals are atomic formulas or negated atomic formulas). The negotiation language differs from Van Veenen and Prakken (2006), where *content* is expressed as a conjunction of literals. A condition can be used to state an action (not) to be performed by the requesting party. For instance, an offer "I will give you the found intelligence data about Soprano on the condition that the data will not be exchanged with other CIEs" can be decomposed into the action "I will give you the found intelligence data about Soprano" and the condition "the data will only be used for analysis purposes".

The following table specifies the negotiation communication language with the reply structure of Van Veenen and Prakken (2006).

Table 2: Negotiation communication language (L_{cn})

speech acts	attacks	surrenders
request(*A, C*)	*offer*(*A', C'*) (*A* ≠ *A'*) *reject*(*A', C'*) (*A* ≠ *A'*)	*withdraw* *accept*(*A, C*)
offer(*A, C*)	*offer*(*A, C'*) (*C* ≠ *C'*) *reject*(*A, C*)	*withdraw* *accept*(*A, C*)
reject(*A, C*)	*offer*(*A, C'*) (*C* ≠ *C'*)	*withdraw*
accept(*A, C*)		
withdraw		

The idea of Table 2 is that *request(A, C)* is an initial request for an offer for the other party to do something (possibly with a condition to-be accepted by the requesting party). Here *A* and *C* are partially uninstantiated and the *offer(A', C')* speech act is a fully instantiated proposal (see Example 4). The reason that a request has free variables is that an agent making a request for an offer does not know which (if any) data unit matches the request. Therefore, when a responding agent offers his found data unit, he instantiates the free variables with data-unit identifier referring the found data unit (data-unit identifiers were described in subsection 6.1.1 on page 125). By means of the *withdraw* speech act an agent withdraws from a negotiation (see Example 1). With *accept(A, C)* an agent accepts an offer and with *reject(A, C)* an agent rejects a request or offer made by another agent. At first sight, the negotiation language is simple and the distinction between attacking and surrendering replies would seem to make no sense. However, the tables specifying the negotiation and persuasion languages will be extended and combined in a way that makes this distinction sensible for negotiation also.

The move notation specified in Definition 3 (page 65) is below for readability purposes slightly changed. Moves are of the form *p*(*S*, *R*, *A*, *C*), where the performative *p* is written outside the move, the identifier and target of a move are left implicit and the other player in the dialogue is added. In the altered notation, the elements *S, R*, *A, C* denote the sender, the receiver, the action and the condition. For instance, with Definition 3, the request to exchange data about Artie Bucco from agent p can be represented as:

```
(1, p, request(exchangeData(o, p, bucco, DATA_UNIT_ID),), 0)
```

In the new notation, the request will be represented as:

```
p1:     request(p, o, exchangeData(o, p, bucco, DATA_UNIT_ID),)
```

In the example, '`request`' is the performative, `p` denotes the sender, `o` denotes the receiver, '`exchangeData(o, p, bucco, DATA_UNIT_ID),`' denotes the action and the condition is empty. Furthermore, p_1 precedes the move to denote that the first move of the dialogue is made by player `p`.

Let us now look at two examples (Example 4 and Example 5) of negotiation dialogues to illustrate the negotiation language specified within the ASPIC logic.

Example 4 no data found about Artie Bucco

In this example, agent `p` requests agent `o` to give him intelligence data about Bucco. However, agent `o` cannot offer anything since he has no intelligence data about Artie Bucco. The initial move starting the dialogue can be represented as:

p_1: `request(p, o, exchangeData(o, p, bucco, DATA_UNIT_ID),)`

Since the responding agent o cannot find intelligence data about Artie Bucco, he cannot offer him the requested data. A possible way to model the response from an agent who cannot find intelligence data is doing a *reject* move. For example '`reject(p, o, exchangeData(o, p, bucco, DATA_UNIT_ID),)`'. However, since only request speech acts can contain uninstantiated terms this is not syntactically correct. This is why a *withdraw* speech act has been added as a possible reply to the negotiation sublanguage of Van Veenen and Prakken (2006) indicating that no intelligence data is found. Therefore the response in this example is:

o_2: `withdraw`

Example 5 requested data about Irina Peltsin granted under a condition

In this example, agent `p` requests agent `o` to give him intelligence data about Irina Peltsin. Agent `o` finds intelligence data about Peltsin and offers to give him this data under the condition that agent p does not pass on the data to other agents.

In move p_1, the data-unit identifier '`DATA_UNIT_ID`' is uninstantiated, which can be used by agents to refer to a data unit[17].

p_1: `request(p, o, exchangeData(o, p, peltsin, DATA_UNIT_ID),)`

In move o_2 the speech act is fully instantiated; the responding agent `o` offers the found data-unit, identified by '`1`' about Peltsin. Furthermore, the responding agent `o`

[17] A data unit is a grouped piece of data, which has a data-unit-id as its associated identifier (see subsection 6.1.1).

states the condition that agent p does not pass on the data to other agents, denoted by '`holds(not(passOn(p, 1)))`'.

```
o2:     offer(o, p, exchangeData(o, p, peltsin, 1),
        holds(not(passOn(p, 1))))
```

Finally, in move o_3 the requesting agent accepts the offer, which terminates the negotiation dialogue.

```
p3:     accept(p, o, exchangeData(o, p, peltsin, 1),
        holds(not(passOn(p, 1))))
```

In the persuasion communication language specified in Table 3, *argue* speech acts state arguments in the ASPIC logic $\boldsymbol{L}$ and P is a formula from the topic language L_t. Furthermore, in Table 3, *argue*(*Arg*') defeats *argue*(*Arg*) according to $\boldsymbol{L}$. Note that a difference between the persuasion communication language of Van Veenen and Prakken (2006) is the addition of the *deny*(*P*) speech act. The *deny*(*P*) speech act is an attacker on the *argue*(*Arg*) speech act and allows expressing that a premise cannot be accepted (agree to disagree). *Deny*(*P*) is similar to *why*(*P*) in that they both are replies on *argue*(*Arg*) speech acts. However, *why*(*P*) allows for further discussion while *deny*(*P*) does not.

Table 3: Persuasion communication language (L_{cp})

speech acts	**attacks**	**Surrenders**
claim(*P*)	*why*(*P*)	*concede*(*P*)
why(*P*)	*argue*(*Arg*) (*conc*(*Arg*) = *P*)	*retract*(*P*)
argue(*Arg*)	*why*(*P*) ($P \in$ *prem*(*Arg*)) *argue*(*Arg*') *deny*(*P*) ($P \in$ *prem*(*Arg*))	*concede*(*P*) ($P \in$ *prem*(*Arg*) or $P =$ *conc*(*Arg*))
concede(*P*)		
retract(*P*)		
deny(*P*)		

Table 4 shows the combination of the two communication languages. The idea of Van Veenen and Prakken (2006) is to add a speech act to the negotiation language that triggers a persuasion dialogue: a new attacking reply is added in L_{cnp} to a *reject*(*A, C*) speech act, namely *why-reject(A, C)*. The only possible reply to this speech act is *claim*(*P*), where *P* is a ground for the rejection. This claim starts a persuasion dialogue.

Table 4: Combined communication language (L_{cnp})

Speech acts	attacks	surrenders
request(*A, C*)	*offer*(*A', C'*) ($A \neq A'$) *reject*(*A', C'*) ($A \neq A'$)	*withdraw*
offer(*A, C*)	*offer*(*A, C'*) ($C \neq C'$) *reject*(*A, C*)	*withdraw* *accept*(*A, C*)
reject(*A, C*)	*offer*(*A, C'*) ($C \neq C'$) *why-reject*(*A, C*)	*withdraw*
accept(*A, C*)		
withdraw		
why-reject(*A, C*)	*claim*(*P*)	
claim(*P*)	*why*(*P*)	*concede*(*P*)
why(*P*)	*Argue*(*Arg*) (*conc*(*Arg*) = *P*)	*retract*(*P*)
argue(*Arg*)	*why*(*P*) ($P \in$ *prem*(*Arg*)) *argue*(*Arg'*) *deny*(*P*) ($P \in$ *prem*(*Arg*))	*concede*(*P*) ($P \in$ *prem*(*Arg*) or $P =$ *conc*(*Arg*))
concede(*P*)		
retract(*P*)		
deny(*P*)		

4.2.2 Communication protocols

In this subsection the protocols for negotiation and persuasion and their combination is defined. There are several design choices for protocols (Prakken 2005): a player can make one move or several moves before the turn shifts (*unique-move vs. multi-move protocols*); a player can make only one or try several alternative replies to a move (*unique-reply vs. multi-reply protocols*); and whether the turn shifts after a player is on the winning side (*immediate vs. non-immediate reply protocols*). Strict protocols are more efficient by constraining the possible moves agents are allowed to make. However, the efficiency comes at the price that not everything can be said. For example, if an agent learns something new during a dialogue and wants to make an alternative reply to a previous turn, then a unique-reply protocol forbids this move. In liberal protocols an agent can make any move as a reply to some earlier move based on L_c, and the turn shifts when the other agent succeeds in making his move(s). Therefore, in the present research a liberal protocol is used. The protocol of this research is multi-reply since it gives agents the freedom to make alternative replies; the protocol is multi-move because during embedded persuasion, agents can make several moves before the turn shifts. The

notion of who is on the winning side is used in the protocol of this research in determining who will reply immediate after terminating an embedded persuasion. Thus the protocol of this research is immediate-reply.

The *negotiation protocol* is straightforward. The requesting agent begins with a *request*, and then the agents take turns after each move, replying to the last move of the other party. Thus negotiations can be of arbitrary length and terminate after an *accept* or a *withdraw* move, since those speech acts have no replies. Negotiations are not guaranteed to terminate.

The *persuasion protocol* is more involved, and is defined as follows.

Definition 4 Protocol for persuasion language L_{cp}

For all dialogues d and moves m it holds that $m \in Pr(d)$ if and only if m satisfies all of the following rules:

- R_6: m satisfies $R_1 - R_5$ of Definition 4 where in R_2 L_c is replaced by L_{cp};
- R_7: if $d = d_0$, then $s(m)$ is of the form $claim(P)$;
- R_8: if m concedes the conclusion of an argument moved in m', then m' does not reply to a *why* move;
- R_9: if the last move of d is a retraction by the proponent or a concession by the opponent of the initial claim of d, then $Pr(d) = \varnothing$.

Rule R_6 particularizes the general structure of the protocol to the persuasion protocol. R_7 states that each persuasion dialogue starts with a claim. The initial claim is the *topic* of the dialogue. R_8 restricts concessions of an argument's conclusion to conclusions of counterarguments. Rule R_8 ensures that propositions are conceded at the move in which they were introduced. R_9 formalizes the idea that a persuasion dialogue terminates when the proponent has retracted his initial claim or when the opponent has conceded the initial claim.

Definition 5 Turntaking function T for persuasion language L_{cp}

- $T: M^{<\infty} \rightarrow Pow(\{P,O\})$

where a dialogue is always started with player P, therefore $T(\varnothing) = \{P\}$. The turntaking function T allows both players to take turns. However, since dialogue $M^{<\infty}$ is a sequence of moves, players cannot make moves at the same time.

Definition 6 Winning criterion for persuasion language L_{cp}

For any dialogue *d*, the proponent is the winner if the opponent has conceded the initial claim, and the opponent is the winner if the proponent has retracted his initial claim.

Note that agents are trusted not to abuse the winning criterion from Definition 6 by never giving in and thus avoid losing. To illustrate how the winning criterion can be abused, consider the following start of a dialogue:

```
p1:     claim c(1)

o2:     why c(1)

p3:     c(1) since c(X) <- b(X). b(1).
```

Let us assume that `o`'s knowledge base contains the following rule and factual statement:

```
{ c(X) <- b(X). b(1).}
```

When player `o` can be trusted, he will concede premise `b(1)` of the argument and also concede `c(1)`, because `b(1)` and `c(1)` can be derived from his knowledge base. Since `c(1)` is the initial claim, and the opponent o concedes to this claim, proponent `p` is the winner according to Definition 6. However, when player `o` cannot be trusted, he can avoid losing by repeating *why* moves. For example,

```
O4:     why b(1)

P5:     b(1) since b(1)
```

At this stage, player `p` gives a circular argument, which in the present research is used to express that `b(1)` is a factual statement from `p`'s knowledge base (see section 5.2). Also in response to move `P5`, player `o` can avoid losing the dialogue by repeating the `why b(1)` move, and thus avoid losing by not conceding to the initial claim:

```
O6:     why b(1)
```

In the present research the agents are given dialogue policies which prevent agents to abuse the winning criterion from Definition 6. The dialogue policies (see chapter 5) prescribe the best choices from the allowed moves specified by the

communication protocol, and the agents are trusted to apply the given dialogue policies.

As for the *combined protocol*, the main idea is that if a negotiation dialogue shifts to a persuasion dialogue, their relation is one of embedding (see McBurney and Parsons 2002): the embedded persuasion dialogue is undertaken until its termination, after which the embedding negotiation dialogue is resumed. So whenever a persuasion move is allowed by the protocol, no negotiation move is allowed. In addition, the structural rules of the persuasion system now also hold for negotiation. This allows for alternative explanations for rejections and for accepting an offer (perhaps conditionally) that was first rejected for reasons that could not be upheld in a persuasion dialogue.

The combined protocol is defined as follows.

Definition 7 Protocol for combined negotiation and persuasion language L_{cnp}

For all dialogues d and moves m it holds that $m \in Pr(d)$ if and only if m satisfies all of the following rules.

- R_{10}: m satisfies $R_1 - R_8$ of Definition 4 and 5 where in R_2 L_c is replaced by L_{cnp}, and in R_6 L_{cp} is replaced by $L_{cnp,}$ in R_7 *claim*(P) replaced by *request*(A, C);
- R_{11}: if the last move of d is a retraction by the proponent or a concession by the opponent of the last claim of d, then s(m) is from L_{cn};
- R_{12}: if $s(m_1)$ = *request*(A, C) and $s(m)$ = *offer*(A', C') or $s(m)$ = *reject*(A', C') then A' and C' do not contain any free variables;
- R_{13}: if $s(m)$ = *offer*(A, C) then no $m' \in d$, $s(m')$ = *offer*(A, C), where m' is unequal to m;
- R_{14}: if $s(m)$ = *offer*(A, C) or *reject*(A, C) or *accept*(A, C) then A and C contain no free variables;
- R_{15}: if the last move m' of d is a persuasion move and there is no move $m \in Pr(d)$ such that $s(m)$ is a persuasion move, then the player who retracts or concedes to the last claim of d, makes an alternative reply to the last negotiation move in d, according to L_{cn}; else if m' is a negotiation move then m replies to m';
- R_{16}: if m is a negotiation move then there is no move $m' \in Pr(d)$ such that $s(m')$ is a persuasion move.

Rule R_{10} generalizes the general structure of the persuasion protocol to the combined protocol and says that each combined dialogue starts with a request for an offer. R_{11} is a replacement for R_9, which states that when an embedded persuasion dialogue terminates, the next move must be a negotiation move which is made by the player who has lost the embedded persuasion. Note that every embedded persuasion dialogue starts with a claim; therefore the last claim of a dialogue is also the last persuasion. Rules R_{12} - R_{14} formalize the negotiation protocol rules of Wooldridge and Parsons (2000) that are not implied by R_{10}. Protocol rule R_{15} prevents unnecessary negotiation backtracking moves. Finally, rule R_{16} performs a key role in the embedding of persuasion in negotiation. R_{16} enforces that the relation between the negotiation and persuasion parts of dialogues is one of embedding of the latter in the former (cf. McBurney and Parsons 2002): as long as a persuasion move is legal, no negotiation move is legal.

Definition 8 Turntaking function *T* for combined negotiation and persuasion language L_{cnp}

$T(d_0) = P;$

$T(d) = O$ if the last move m' of dialogue d, $s(m')$ is a negotiation move and $pl(m') = P$;

$T(d) = P$ if the last move m' of dialogue d, $s(m')$ is a negotiation move and $pl(m') = O$;

$T(d) = \{P,O\}$ if the last move m' of dialogue d, $s(m')$ is a persuasion move and there is at least one move $m \in Pr(d)$ such that $s(m)$ is a persuasion move;

$T(d) = P$ if the last move m' of dialogue d, $s(m')$ is a persuasion move and $Pr(d) = \varnothing$ and player P has lost the persuasion dialogue;

$T(d) = O$ if the last move m' of dialogue d, $s(m')$ is a persuasion move and $Pr(d) = \varnothing$ and player O has lost the persuasion dialogue;

In the combined protocol, agents take turns after each negotiation move and during embedded persuasion dialogues both agents are free to make take turns after the initial claim. In embedded persuasion dialogues turntaking is left to the agents, since they are trusted not to abuse their freedom given by the liberal protocol. As for turntaking a after a terminated embedded persuasion dialogue, the next negotiation move must be made by the player who has lost the embedded persuasion. In (Prakken 2005; Van Veenen and Prakken 2006) the flexibility of protocols is constrained by the use of the notion of *relevance*. Relevance is defined in terms of the notion of dialogical status of a move. Briefly, a move is *in* if it has a surrendering reply or else all its attackers are *out*; and a move is *out* if is it not surrendered and has an attacking reply that is *in*, see Prakken (2005).

Example 6 terrorist assault on the queen

Let us illustrate the combined negotiation and persuasion protocol with an example. In this scenario, which is based on Example 2 (page 34), the requesting agent p requests the responding agent o to offer him intelligence data about a terrorist group in Amsterdam which is possibly planning an assault on the queen. Initially the responding agent rejects his request. However, the requesting agent persuades him to exchange the requested data. Note that this example assumes that both agents have a knowledge base and that only legal moves to previous moves are shown, unless backtracking is relevant. The dialogue starts with the request from agent p to agent o to give him intelligence data about a possible terrorist assault:

p_1: `request(p, o, exchangeData(o, p, terrorists, DATA_UNIT_ID),)`

This is the only legal move type according to protocol rule R_{10} combined with R_7 which state that dialogues start with a request move.

Based on protocol rule R_{10} combined with rule R_2, the next legal replies are: to offer the found data, or to reject the request, or to withdraw from the dialogue. Assuming that the responding agent o finds matching data, he chooses to reject the request. Furthermore, the replying move satisfies rule $R12$. *Protocol rule* $R12$ states that the free variables from the request must be instantiated, i.e., `DATA_UNIT_ID` is instantiated with `3`.

o_2: `reject(o, p, exchangeData(o, p, terrorists, 3),)`

The allowed reply moves (based on R_{10} combined with R_2) to a reject are making a counteroffer with another condition or making a *why-reject* move. Agent p chooses a *why-reject* move to try to persuade the responding agent o. Therefore the agent p states 'why don't you give me that data?'

p_3: `why-reject(p, o, exchangeData(o, p, terrorists, 3))`

Based on R_{10} combined with R_2 the only legal reply to a *why-reject* move is a claim (see the combined negotiation and persuasion language L_{cnp} specified in Table 4 on page 70). Therefore the agent claims that it is forbidden to exchange data about a possible terrorist assault.

o_4: `claim(o, p, obliged(not(exchangeData(o, p, terrorists, 3)))`

The claim is the start of an embedded persuasion dialogue. Protocol rule R_{16} states that only persuasion language moves are allowed during persuasion.

According to R_{10} combined with R_2, the legal replies to a claim are making a *why* or *concede* move. Agent p chooses to do a *why* move and asks 'why is it forbidden to exchange that data?'.

p_5: `why(p, o, obliged(not(exchangeData((o, p, terrorists, 3))))`

The allowed replies to a *why* move are retracting the claim or giving an argument for the claim (based on R_{10} combined with R_2). Agent o gives an argument 'it is forbidden to exchange the data since the data is too old' for the claim.

o_6:
```
argue(o, p,
obliged(not(exchangeData(o, p, terrorists, 3))).

obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID))
<-
holds(dataOlderThanOneYear(DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)).

holds(dataOlderThanOneYear(3)).
requested(exchangeData(o, p, terrorists, 3)).
valid(rOlderThanOneYear_F(o, p, terrorists, 3))).
)
```

Assuming that agent p checks the conclusion and the premises of the argument, the requesting agent p makes several moves at this turn in the dialogue. The legal reply moves to an argument are conceding the conclusion or premises of the argument, denying premises of the argument, making *why* moves targeted at premises of the argument or creating a counterargument.

The argument is constructed using the '`rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)`' domain rule. The agent states the argument that the rule is not applicable in this case:

p_7-o_6:
```
argue(p, o,
~rOlderThanOneYear_F(o, p, terrorist, 3).

~rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)
<-
holds(exceptionalSituation(DATA_UNIT_ID)),
valid(rExceptionalSituation_OTOY(S, R, QUERY, DATA_UNIT_ID)).

holds(exceptionalSituation(3))
valid(rExceptionalSituation_OTOY(o, p, terrorists, 3)).
)
```

The agent decides to continue with the premises of the argument.

The first premise of the argument is '`holds(dataOlderThanOneYear(3))`'. This is an unknown premise for agent `p`. Therefore the agent asks why.

p_8-o_6: `why(p, o, holds(dataOlderThanOneYear(3)))`

The second premise of the argument is '`requested(exchangeData(o, p, terrorists, 3))`' . Assuming that the agent can derive this premise from his knowledge base, he concedes the second premise.

p_9-o_6: `concede(p, o, requested(exchangeData(o, p, terrorists, 3))`

Although protocol rule R_8 invalidates conceding to the conclusion of the argument, this concede is targeted at a premise of the argument and is therefore legal.

p_{10}-o_6:`concede(valid(rOlderThanOneYear_F(o, p, terrorists, 3)))`

At this turn, the responding agent checks the incoming moves p_7, p_8, p_9 and p_{10}.

Move p_7 is an argument, and similar to the reaction to move o_6, agent `p` chooses to continue with the premises of the argument.

The first premise of the argument is '`holds(exceptionalSituation(3))`'. This is an unknown premise for agent `o`. Therefore the agent asks why.

o_{11}-p_7:`why(o, p, holds(exceptionalSituation(3)))`

The second premise of the argument is '`valid(rExceptionalSituation_OTOY(o, p, terrorists, 3))`', which the agent chooses to concede.

o_{12}-p_7: `concede(o, p, valid(rExceptionalSituation_OTOY(o, p, terrorists, 3)))`

In response to move p_8, agent `p` gives a trivial circular argument indicating a factual statement 'the data is older than one year because it is older than one year'.

```
o13-p8:argue(o, p,
        holds(dataOlderThanOneYear(3)).

        holds(dataOlderThanOneYear(3)).
        )
```

Since no replies are defined on a *concede* move (see Table 4 on page 70), agent p does not reply to move p_9.

At this turn the requesting agent p checks the incoming moves o_{11}, o_{12} and o_{13}.

Based on protocol rule R_{10} combined with R_2, the allowed replies to the *why* move o_{11} are retracting the premise of the argument or giving an argument for the premise. Agent p decides to gives an argument for the premise 'there is an exceptional situation because there is a possible attempted assault on the queen'.

```
p14-o11:argue(p, o,
        holds(exceptionalSituation(3)).

        holds(exceptionalSituation(DATA_UNIT_ID))
        <-
        holds(attemptAssaultQueen(DATA_UNIT_ID)),
        valid(rQueen_ES(DATA_UNIT_ID)).

        holds(attemptAssaultQueen(3)).
        valid(rQueen_ES(3)).
        )
```

Move o_{13} is a trivial circular argument indicating a fact, assuming agent p trusts agent o, he chooses to accept the premise.

```
p15-o13:concede(p, o, holds(dataOlderThanOneYear(3)))
```

At this turn the responding agent checks the incoming moves p_{14} and p_{15}. Agent o does not reply to move p_{15} since no replies are defined on a *concede* move (see Table 4 on page 70).

Move p_{14} is an argument and agent o checks the conclusion and the premises of the argument. Assuming that the agent cannot concede the conclusion of the argument and also cannot create a counterargument or undercut for the argument, he continues with the premises of the argument.

The first premise of the argument is '`holds(attemptAssaultQueen(3))`'. This is an unknown premise for agent o. Therefore the agent asks why.

o_{16}-p_{14}:why(o, p, holds(attemptAssaultQueen(3)))

The second premise of the argument is 'valid(rQueen_ES(3))'. Assuming that the agent can derive this premise from his knowledge base, he concedes the second premise.

o_{17}-p_{14}:concede(o, p, valid(rQueen_ES(3)))

At this turn the requesting agent checks moves o_{16} and o_{17}. In response to move o_{16}, agent p gives a trivial circular argument 'there is an attempted assault on the queen because there is an attempted assault on the queen':

p_{18}-o_{16}:argue(p, o,
holds(attemptAssaultQueen(3)).

holds(attemptAssaultQueen(3)).
)

Since no replies are defined on a *concede* move, agent p does not reply to move o_{17}.

At this turn responding agent o checks move p_{18} which is a trivial circular argument. Assuming that agents trust each other, he chooses to concede the premise.

o_{19}-p_{18}:concede(o, p, holds(attemptAssaultQueen(3)))

By conceding to move p_{18}, agent o's original reason for 'obliged(not(exchangeData(o, p, terrorists, 3))' is now defeated and therefore o concedes to p's negotiation: 'I concede that in this case it is not forbidden to exchange the data, so I retract that it is forbidden to exchange the data'.

o_{20}-p_{5}:retract(o, p, obliged(not(exchangeData(o, p, terrorists, 3)))

With move o_{20} the responding agent o backtracks to move p_5, now surrendering to that move. Agent o has retracted his initial claim of the embedded persuasion. Therefore, according to protocol rule R_{11} the persuasion is now terminated and the dialogue shifts back to the negotiation dialogue. Based on protocol rule R_{16} the legal next move must be a negotiation move. Furthermore, R_4 requires that if a player backtracks, the new move must be different from the first one.

Based on protocol rule R_{15} the legal next move must be an alternative reply to the last negotiation move made by agent o (since he has retracted the last claim). Therefore after move o_{20} agent o backtracks to move p_1. At this moment agent o infers that it is not forbidden to exchange the requested data about terrorist attacks. Therefore, he chooses to offer the requested data.

o_{21}-p_1: offer(o, p, exchangeData(o, p, terrorists, 3)))

The replying move satisfies rule R_{12} which states that the free variables from the request must be instantiated, i.e., DATA_UNIT_ID is instantiated with 3.

In response to move o_{21}, the requesting agent accepts the offer.

p_{22}-o_{21}:accept(p, o, exchangeData(o, p, terrorists, 3)))

Move p_{22} terminates the dialogue since agent o must reply to agent p's last move and no reply moves are defined on *accept* moves (see L_{cnp} specified in Table 4 on page 70).

4.3 Chapter summary

This chapter has specified the argumentation system to model the internal inferences of an agent and the dialogue system to model the interaction between agents. Section 4.1 presented ASPIC as a logic for defeasible argumentation and as knowledge representation language. As discussed in section 3.3 (page 47), agents especially reason about whether something is obligatory, forbidden, or whether something violates their interest. The deontic modalities are not part of the ASPIC logic. However, they are expressed with reification. Also the agent goals (see section 3.2 on page 45) are expressed with reification. Section 4.2 specified the dialogue system for the interactions between agents. In the police domain, the agent interactions are interpreted as negotiation with embedded persuasion dialogues, and this is precisely the kind of embedding modeled in Van Veenen and Prakken (2006). Therefore, the chosen language and protocol is based on their work. To be compatible with the ASPIC inference engine and to support the regulated data exchange between CIE officers the dialogue systems are adapted in several ways. The negotiation language differs from Van Veenen and Prakken (2006), where the content of a speech act is expressed as a conjunction of literals. While in the present research the elements of *content* are literals *A* and *C*, which denote the action and the condition. The condition can be used to state an action (not) to be performed by

the requesting party. A second difference is the addition of the *deny*(*P*) speech act and the *withdraw* speech act as a reply to a request to the communication languages of Van Veenen and Prakken (2006). The present research uses liberal protocols to allow agents to explore alternative moves. In (Prakken 2005; Van Veenen and Prakken 2006), excessive flexibility of protocols is constrained by the use of the notion of relevance. In this research, the notion of relevance is not required for the protocols, since CIE agents are trusted not to abuse their freedom given by liberal protocols. The next chapter will specify and formalize policies for negotiation and persuasion dialogues between agents.

Chapter 5

Dialogue policies

The previous chapter specified the argumentation system to model the internal inferences of an agent and the dialogue system to model the interaction between agents. This chapter presents the policies for negotiation and persuasion dialogues between agents[18]. First the idea of policies for dialogue systems is presented. Then in section 5.1 and section 5.2, the policies for negotiation and for persuasion are described. Finally in section 5.3 the application of the policies in the Dutch police domain is discussed and an example is presented to illustrate the policies.

Whereas a communication protocol specifies all the allowed moves, a policy specifies the best choices from those moves. In case of dialogues between CIE agents about regulated data exchange, the best move is one that is expected to further the goal to execute the appointed police task and the goal to protect local investigations and informants, while incorporating the attitude of the criminal investigation unit. The attitude of CIEs is reflected in their application of usage codes to data and their willingness to exchange protected data. For example, a criminal investigation unit with an overcautious attitude labels data as protected more often than required. Furthermore, an overcautious criminal investigation unit is mainly focused on the goal to protect local data and therefore will not easily provide data, while a cooperative criminal investigation unit is more easily persuaded to exchange protected data.

For the proof-of-concept application, two types of dialogue policies need to be specified, viz. for negotiation and for persuasion. A responding agent considers the normative issue whether performing a requested action is obligatory or forbidden, and the teleological issue whether performing an action is a violation of his interests. A requesting agent considers the normative issue whether agreeing with a condition stated in an offer is obligatory or forbidden, and the teleological issue

[18] Note that this chapter is based on earlier work published in (Dijkstra et al. 2005; Dijkstra et al. 2006; Dijkstra et al. 2007).

whether agreeing is a violation of his interests. Persuasion policies determine, for example, what kinds of arguments (e.g., defensible, justified) an agent should accept, when it is allowed to give a counterargument to an argument moved by the other agent and whether a claim or argument can be accepted. The agents' policies should induce a fair degree of cooperativeness since the policies are designed to further the agent's goals and these goals include those of the overall organization.

The policies use the ASPIC reasoner (see subsection 4.1.1) to determine whether it is possible to create a justified or defensible argument for a given conclusion. As for notation**,** the style of Karacapilidis and Moraitis (2002) ' $[A] \vdash_{\text{status-type}} B$' is adopted to denote that a reasoner is called to infer that the set of rules A implies conclusion B, and the status-type denotes if conclusion B is defensible or justified. The ASPIC language does not contain double negation; therefore the occurrence of *not*(A), when A is a negated formula, should strictly speaking be replaced with A. For simplicity, this complication is left implicit. Finally, the policies assume a selection mechanism which chooses the moves to be replied to in a dialogue.

5.1 Negotiation policies

This section presents the specification of the negotiation policies. First the negotiation policies for (A) *offer*, (B) *request*, (C) *reject* and (D) *why-reject* speech acts are formalized. Then an example is given to illustrate the negotiation policies. As specified in subsection 4.2.1 (page 66) the negotiation communication language consists of *speech acts p(content)* where p is a performative and *content* is defined in topic language L_t with associated logic $\boldsymbol{L}$. The elements of *content* are literals A and C, which denote the action and the condition. Furthermore, moves are of the form $p(S, R, A, C)$ where the performative of the speech act p is written outside the move, the identifier and target of a move are left implicit and the other player in the dialogue is added. The elements of a move are p to denote the performative and S, R, A, C denote the sender, the receiver, the action and the condition. Finally, C is of the form 'holds(D)' or 'happens(D)'.

A: Reacting to an offer

When an agent is in a responding role and receives an offer, he deliberates whether it is obliged, forbidden or whether it is a violation of its own interests to perform the action specified in the offer. If the agent concludes that it is obliged to perform the action, he accepts the offer. If he concludes that it is forbidden to perform the action, then he rejects the offer. If the agent concludes that it is permitted to perform the action but there is a violation of the agent's own interests then he tries to find a condition under which there will be no violation of its own interests. If he

has succeeded in finding such a condition, he makes a counteroffer with the condition. If he has not succeeded in finding a condition, he rejects the offer.

When a requesting agent receives an offer, he deliberates whether it is obliged, forbidden or whether it is a violation of his own interests to agree with the condition stated in the offer. The agent accepts the offer when he concludes that it is obliged to agree with the condition. The agent rejects the offer when he concludes it that is forbidden to agree with the condition. If the agent concludes that it is permitted to agree with the condition but there is a violation of the agent's own interests then he tries to find a condition under which there will be no violation of his own interests. If he has succeeded in finding such a condition, he makes a counteroffer with the extra conditions, else he rejects the offer.

```
IF selectedMove(IncomingMove) = offer(S, R, A, C)
THEN
```

Agents have different negotiation policies for an offer depending on their role. Therefore, the first step of the negotiation policy for an offer is to determine the agent role, which in this case is the role of responding agent.

```
1.1.
     IF agentRole(R) = responding
```

Step 1.1.1 of the negotiation policy for an offer checks whether it is possible to create a justified or defensible argument (*status-type*) for the conclusion that the requested action is obliged. If that is possible, the negotiation policy returns the answer to accept the requested content. The condition *C,* which is the action to be performed by the other requesting agent *S,* is conjoined with the knowledge base to determine whether it is obliged to offer the requested data. Moreover, another thing which has to be added to the knowledge base is *requested*(*A*), since determining whether an action is obliged, presupposes that the action is requested. Because of the assumption that the agents trust each other (see section 2.4 on page 18), the requesting agent is trusted to determine whether the condition can be met.

```
1.1.1.
      IF [KB_argumentation-engine ∪ {requested(A)} ∪ {C}]
      ⊢_status-type obliged(A)
      THEN
                OutgoingMoves <- OutgoingMoves + accept(R, S, A, C)
```

The policies assume a selection mechanism to choose the moves to be replied to in a dialogue, where each chosen move serves as input for the policies. For each applicable policy step, the policy stores the reply move in *OutgoingMoves*. When every applicable policy step is taken, the policy returns all the reply moves contained in *OutgoingMoves*.

The negotiation policy for an offer could reject the request for data if there is no justified argument which says that it is obliged to exchange the found data.

However, this action is not in line with the goal of CIEs to execute their appointed police task by exchanging data (see section 2.6 on page 37). Therefore, in the next negotiation policy steps, the policy tries to exchange data such that the requested data is not forbidden to exchange and the agents' own interests are not violated.

Step 1.1.2 of the negotiation policy checks whether it is possible to create a justified or defensible argument for the conclusion that the requested action is forbidden. If that is possible, the negotiation policy returns the answer to reject the offer. Since only the 'obliged' operator is used (see subsection 4.1.2 on page 61), 'forbidden(*A*)' is rewritten as 'obliged(not(*A*))'.

```
1.1.2.
     ELSEIF [KB_argumentation-engine ∪ {requested(A)} ∪ {C}]
     ⊢_status-type obliged(not(A))
     THEN
               OutgoingMoves <- OutgoingMoves + reject(R, S, A, C)
```

Step 1.1.3 of the policy checks whether it is possible to create a justified or defensible argument for the conclusion that the offer is a violation of the agent's (denoted by *R*) own interests. If that is impossible, the negotiation policy returns the answer to accept the offer. Moreover, another thing which has to be added to the knowledge base is *happens*(*A*), instead of *requested*(*A*). Since determining whether an action is a violation of interests presumes that the action happens.

```
1.1.3.
     ELSEIF [KB_argumentation-engine ∪ {happens(A)} ∪ {C}]
     ⊬_status-type violationOfOwnInterests(R)
     THEN
               OutgoingMoves <- OutgoingMoves + accept(R, S, A, C)
```

In step 1.1.4 of the negotiation policy for an offer, the argumentation system tries to find a condition for the conclusion such that the responding agent's own interests are not violated and that the requested action is not forbidden. If that is possible, the negotiation policy returns the answer to make a counteroffer with the new condition *C*' (where *C'* is unequal to every other *C* before). If that is not possible, the negotiation policy returns the answer to reject the offer. Step 1.1.4 of the negotiation policy uses abductive reasoning to find a condition under which a new offer can be made. The ASPIC reasoner is not capable yet of abductive reasoning; therefore in the current implementation this is done manually.

```
1.1.4.
     ELSEIF there exists a C' such that
     [KB_argumentation-engine ∪ {happens(A)} ∪ {C'}]
     ⊬_status-type violationOfOwnInterests(R)
     ∧ [KB_argumentation-engine ∪ {requested(A)} ∪ {C'}]
     ⊬_status-type obliged(not(A))
     THEN
               OutgoingMoves <- OutgoingMoves + offer(R, S, A, C')
     ELSE
               OutgoingMoves <- OutgoingMoves + reject(R, S, A, C)
```

Step 1.2 of the negotiation policy for an offer checks if the agent has the role of a requesting agent.

```
1.2.
     IF agentRole(R) = requesting
     THEN
```

Step 1.2.1 of the negotiation policy for an offer checks whether it is possible to create a (justified or defensible) argument for the conclusion that the stated condition *C* is obliged. If that is possible, the negotiation policy returns the answer to accept the offer.

```
1.2.1.
     IF [KB_argumentation-engine ∪ {requested(A)}]
     ⊢_status-type obliged(C)
     THEN
               OutgoingMoves <- OutgoingMoves + accept(R, S, A, C)
```

Step 1.2.2 of the negotiation policy checks whether it is possible to create a (justified or defensible) argument for the conclusion that the stated

condition *C* is forbidden. If that is possible, the negotiation policy returns the answer to reject the offer.

```
1.2.2.
     ELSEIF [KB_argumentation-engine ∪ {requested(A)}]
     ⊢_status-type obliged(not(C))
     THEN
               OutgoingMoves <- OutgoingMoves + reject(R, S, A, C)
```

Step 1.2.3 of the policy checks whether it is possible to create a justified or defensible argument for the conclusion that the requested action with the stated condition is a violation of the agent's (denoted by *R*) own interests. If that is impossible, the negotiation policy returns the answer to accept the offer.

```
1.2.3.
     ELSEIF [KB_argumentation-engine ∪ {happens(A)} ∪ {C}]
     ⊬_status-type violationOfOwnInterests(R)
     THEN
               OutgoingMoves <- OutgoingMoves + accept(R, S, A, C)
```

Note that determining whether accepting a condition causes a violation of interests presumes that the condition 'holds' or 'happens'. Therefore the condition is conjoined with the knowledge base.

In step 1.2.4 of the negotiation policy for an offer, the argumentation system tries to find a new condition such that the requesting agent's own interests are not violated and that the new condition is not forbidden. If that is possible, the negotiation policy returns the answer to make a counteroffer with the new condition *C'* (where *C'* is unequal to every other *C* before). If that is not possible, the negotiation policy returns the answer to reject the offer.

```
1.2.4.
     ELSEIF there exists a C' such that
     [KB_argumentation-engine ∪ {happens(A} ∪ {C'}]
     ⊬_status-type violationOfOwnInterests(R)
     ∧ [KB_argumentation-engine ∪ {requested(A}]
     ⊬_status-type obliged(not(C'))
     THEN
               OutgoingMoves <- OutgoingMoves + offer(R, S, A, C')
     ELSE
               OutgoingMoves <- OutgoingMoves + reject(R, S, A, C)
```

B: Reacting to a request

A request is a special case of an offer and is only applicable for responding agents. When an agent receives a request and he cannot perform the requested action, he withdraws from the dialogue. If he can perform the requested action, the agent takes the same steps as a requesting agent receiving an offer, i.e., he deliberates whether it is obliged, forbidden or whether it is a violation of his own interests to perform the action specified in the request.

```
2.
    IF selectedMove(Move) = request(S, R, A, C)
    THEN
```

As described in subsection 6.1.2 on page 127, the responding agent first searches for a matching data-unit in his data source when receiving a request. The free variable in the request (data-unit identifier) is instantiated in his offer, depending on whether the responding agent finds a matching data unit. If he succeeds finding a matching data-unit, the free variable is instantiated with a data-unit identifier referring to the found data unit. If he cannot find a matching data unit, the request is instantiated with 'empty' placeholders.

Step 2.1 of the negotiation policy for a request checks if no matching data unit is found. If that is the case (denoted by the 'empty' constants), the policy returns the answer to withdraw.

```
2.1.
     IF request(S, R, A, C) = request(S, R, empty, empty)
     THEN
            OutgoingMoves <- OutgoingMoves + withdraw(R, S, A, C)
```

Step 2.2 is equal to step 1.1.1 of the negotiation policy for an offer.

```
2.2.
     ELSEIF [KB_argumentation-engine ∪ {requested(A)} ∪ {C}]
     ⊢_status-type obliged(A)
     THEN
            OutgoingMoves <- OutgoingMoves + accept(R, S, A, C)
```

Step 2.3 equals step 1.1.2 of the negotiation policy for an offer.

2.3.
ELSEIF [$KB_{argumentation\text{-}engine}$ ∪ {requested(*A*)} ∪ {*C*}]
$\vdash_{status\text{-}type}$ obliged(not(*A*))
THEN
OutgoingMoves <- *OutgoingMoves* + reject(*R*, *S*, *A*, *C*)

Step 2.4 of the policy for a request is equal to negotiation policy step 1.1.3

2.4.
ELSEIF [$KB_{argumentation\text{-}engine}$ ∪ {happens(*A*)} ∪ {*C*}]
$\nvdash_{status\text{-}type}$ violationOfOwnInterests(*R*)
THEN
OutgoingMoves <- *OutgoingMoves* + accept(*R*, S, *A*, *C*)

Step 2.5 is equal to step 1.1.4 of the negotiation policy for an offer.

2.5.
ELSEIF there exists a *C'* such that:
[$KB_{argumentation\text{-}engine}$ ∪ {happens(*A*)} ∪ {*C'*}]
$\nvdash_{status\text{-}type}$ violationOfOwnInterests(*R*)
∧ [$KB_{argumentation\text{-}engine}$ ∪ {requested(*A*)} ∪ {*C'*}]
$\nvdash_{status\text{-}type}$ obliged(not(*A*))
THEN
OutgoingMoves <- *OutgoingMoves* + offer(*R, S, A, C'*)
ELSE
OutgoingMoves <- *OutgoingMoves* + reject(*R, S, A, C*)

C: Reacting to a reject

When an agent receives a rejection, he first determines whether the *reject* move was made earlier in the dialogue. If the reject move was not made earlier to the previous move, he will do a *why-reject* move to challenge the rejection and this *why-reject* move initiates an embedded persuasion dialogue. If the rejection was made earlier, and therefore an embedded persuasion dialogue has occurred during the dialogue, the agent checks if he has lost the embedded persuasion. If that is the case, the agent tries to make an offer with a new condition such that the requested action is not forbidden and does not violate his interests. Else, he withdraws from the dialogue.

3.
IF selectedMove(*Move*) = reject(*S*, *R*, *A*, *C*)
THEN

In step 3.1 the agent first checks if the *reject* move was not made earlier as a reaction to the previous move. If that is the case, the agent returns a *why-reject* move to start an embedded persuasion dialogue.

```
3.1.
    IF the reject move has not yet been replied to before
    THEN
          OutgoingMoves <- OutgoingMoves + why-reject(R, S, A, C)
    ELSE
```

The 'else' part of step 3.1 implies that there has already been an embedded persuasion dialogue which challenged the rejection. Moreover, since the policies assume a selection mechanism to choose the moves to be replied to in a dialogue, the agent infers that reselection of the *reject* move implies that the persuasion has been lost. Therefore, the agent will not start another embedded persuasion dialogue and tries to make an offer with a new condition in step 3.1.1. If he succeeds he makes the offer, else he withdraws from the dialogue.

```
3.1.1.
     IF [KB_argumentation-engine ∪ {happens(A)} ∪ {C'}]
     ⊬_status-type violationOfOwnInterests(R)
     ∧ [KB_argumentation-engine ∪ {requested(A)} ∪ {C'}]
     ⊬_status-type obliged(not(A))
     THEN
               OutgoingMoves <- OutgoingMoves + offer(R, S, A, C')
     ELSE
               OutgoingMoves <- OutgoingMoves + withdraw
```

D: Reacting to a why-reject

When an agent receives a *why-reject* move, he will use the argument he has already constructed for his rejection. The conclusion of the argument is then returned as the claim, which starts an embedded persuasion dialogue. There are two situations when an agent rejects an offer or a request. The first situation is that the proposal was rejected on the basis that the requested action is forbidden. The second situation is that the proposal was rejected on the basis that the requested action is a violation of the agent's interests

```
4.
   IF selectedMove(Move) = why-reject(S, R, A, C)
   THEN
```

In step 4.1 the negotiation policy queries the argumentation system if it can create an (status-type) argument for the conclusion that the action is forbidden. If this is possible, the negotiation policy returns the conclusion as the claim.

```
4.1.
     IF [KB_argumentation-engine ∪ {requested(A)} ∪ {C}]
     ⊢_status-type obliged(not(A))
     THEN
            OutgoingMoves <- OutgoingMoves
            + claim(R, S, obliged(not(A)))
```

In step 4.2 the policy queries the argumentation system if it can create an (*status-type*) argument for the conclusion that the agent's interests are violated. If this is possible, the policy returns the conclusion (that there is a violation of the agent's interests) as the claim.

```
4.2.
     ELSEIF [KB_argumentation-engine ∪ {happens(A)} ∪ {C}]
     ⊢_status-type violationOfOwnInterests(R))
     THEN
            OutgoingMoves <- OutgoingMoves
            + claim(R, S, violationOfOwnInterests(R))
```

Finally, at the end of the previous negotiation steps, the resulting move is returned to the agent calling the negotiation policy.

```
5.
    RETURN OutgoingMoves
```

Example 7 source protection & analysis purposes

This scenario, which is based on Example 1 (page 33), illustrates how a responding agent can state a condition and how a requesting agent can accept an offer with a condition. In this example, the requesting agent p requests the responding agent o to give him intelligence data about Adriana.

p_1: `request(p, o, exchangeData(o, p, adriana, DATA_UNIT_ID),)`

The responding agent o tries to find a matching data-unit identifier for the request (for simplicity, only one data-unit identifier matches the query). The responding agent finds a matching data-unit identifier ('2') and instantiates the move as:

```
request(p, o, exchangeData(o, p, adriana, 2),).
```

Since the selected move is a request, step 2 of the negotiation policy is applicable. First agent o applies Step 2.1 to check if no matching data unit is found; this is not the case. Therefore, o applies step 2.2 of the policy. In step 2.2, the negotiation policy queries the argumentation system whether it can find a justified argument for the conclusion that it is obliged to exchange the found intelligence data.

```
IF [KB_argumentation-engine ∪ {requested(exchangeData(o, p,
adriana, 2))}]
⊢_justified obliged(exchangeData(o, p, adriana, 2))
THEN
        accept(o, p, exchangeData(o, p, adriana, 2),)
```

Since no justified argument is found in step 2.2, agent o continues with step 2.3 of the negotiation policy for a request. Step 2.3 checks whether the argumentation system can find a justified argument for the conclusion that it is forbidden to give the found intelligence data.

```
IF [KB_argumentation-engine ∪ {requested(exchangeData(o, p,
adriana, 2))}]
⊢_justified obliged(not(exchangeData(o, p, adriana, 2)))
THEN
        reject(o, p, exchangeData(o, p, adriana, 2),)
```

Note that '`forbidden(P)`' is substituted with '`obliged(not(P))`', see subsection 4.1.2 on page 61. Also in step 2.3, the argumentation system cannot find a justified argument for the conclusion that is forbidden to exchange intelligence data. Therefore, agent o applies step 2.4 of the negotiation policy. Step 2.4 checks if there is no violation of the agent's own interests if the agent exchanges the requested data.

```
IF [KB_argumentation-engine ∪ {happens(exchangeData(o, p, adriana,
2))}]
⊬_justified violationOfOwnInterests(o)
THEN
        accept(o, p, exchangeData(o, p, adriana, 2),)
```

Based on domain rule *rTraceable_SP* (see page 167), the agent infers that informant Adriana has source protection since the data is traceable to Adriana.

```
[
holds(sourceProtection(DATA_UNIT_ID, INFORMANT))
```

```
<-
holds(traceable(DATA_UNIT_ID, INFORMANT)),
valid(rTraceable_SP(DATA_UNIT_ID, INFORMANT)) 0.3.

holds(traceable(2, adriana)).
valid(rTraceable_SP(2, adriana)).
]

⊢justified
holds(sourceProtection(2, adriana))
```

Furthermore, based on domain rule *rSourceProtection_NP* (see page 171), the agent infers that informant Adriana who has supplied the data, has no protection when the data is exchanged.

```
[
holds(not(protection(DATA_UNIT_ID, INFORMANT)))
<-
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
happens(exchangeData(S, R, ,QUERY, DATA_UNIT_ID)),
valid(rSourceProtection_NP(S, R, ,QUERY,
DATA_UNIT_ID,INFORMANT)).

happens(exchangeData(o, p, ,adriana, 2)),
valid(rSourceProtection_NP(o, p, adriana, 2, adriana)).
]

⊢justified
holds(not(protection(2, adriana)))
```

Based on the *rProtectionGoal* domain rule (see page 171), the agent concludes that a goal for him is to protect informant Adriana who has supplied the data.

```
[
goal(S, protection(DATA_UNIT_ID, INFORMANT))
<-
holds(informantOf(INFORMANT, DATA_UNIT_ID, S)),
valid(rProtectionGoal(DATA_UNIT_ID)).

holds(informantOf(adriana, 2, o)).
valid(rProtectionGoal(o, 2, adriana)).
]

⊢justified
goal(o, protection(2, adriana)))
```

Finally, based on domain rule *rViolationOfOwnInterestsA* (see page 172), the agent infers that his interests are violated since his goal to protect informant Adriana conflicts with the conclusion that Adriana is not protected when the data with associated data-unit identifier `2` is exchanged:

```
[
violationOfOwnInterests(S)
<-
holds(not(P)), goal(S, P).

holds(not(protection(2, adriana)).
goal(o, protection(2, adriana)).
]
```

$\vdash_{justified}$ violationOfOwnInterests(o)

Now the agent continues with step 2.5 of the policy and tries to find a condition under which there is no violation of his interests. By abductive reasoning, agent o finds a condition '`holds(analysisPurposesOnly(p, 2))`.' in domain rule *rSourceProtectionAnalysis_P* (see page 166) . Based on this rule the agent infers that it is permitted to exchange the requested data, if Adriana who has supplied that data has source protection and that the data will be used for analysis purposes only:

```
[
~obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
holds(analysisPurposesOnly(CIE_OFFICER, DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rSourceProtectionAnalysis_P(S, R, QUERY,
DATA_UNIT_ID, CIE_OFFICER)).

holds(sourceProtection(2, adriana)).
holds(analysisPurposesOnly(p, 2)).
requested(exchangeData(o, p, adriana, 2)).
valid(rSourceProtectionAnalysis_P(o, p, adriana, 2, p)).
]
```

$\vdash_{justified}$ ~obliged(not(exchangeData(o, p, adriana, 2)))

therefore step 2.5 of the negotiation policy for a request succeeds:

$[KB_{argumentation\text{-}engine} \cup$ {happens(exchangeData(o, p, adriana, 2))} $\cup$ {holds(analysisPurposesOnly(p, 2))}]

$\nvdash_{justified}$ violationOfOwnInterests(o)

$\wedge$

$[KB_{argumentation\text{-}engine} \cup$ {requested(exchangeData(o, p, adriana, 2))}

$\cup$ {holds(analysisPurposesOnly(p, 2))}]

$\nvdash_{justified}$ obliged(not(exchangeData(o, p, adriana, 2)))

THEN

offer (o, p, exchangeData(o, p, adriana, 2), holds(analysisPurposesOnly(p, 2)))

The negotiation policy for a request is completed and the responding agent o puts the resulting move in the dialogue.

o_2: offer(o, p, exchangeData(o, p, adriana, 2), holds(analysisPurposesOnly(p, 2)))

At dialogue step o_2, agent o offers the requesting agent to give him intelligence data about Adriana under the condition that the data can be used for analysis purposes only. Since agent p receiving the offer is in a requesting role, negotiation policy step 1.2 is applicable. The first step (1.2.1) of the policy is to check whether it is possible to create a justified or defensible argument (*status-type*) that the condition is obliged:

IF [$KB_{argumentation\text{-}engine}$ ∪ {requested(exchangeData(o, p, adriana, 2))}]
$\vdash_{justified}$ obliged(analysisPurposesOnly(p, 2))
THEN
accept(o, p, exchangeData(o, p, adriana, 2), holds(analysisPurposesOnly(p, 2)))

This is possible, based upon domain rule *rAnalysisQuery_AP* (see page 171) which states that if a query from a CIE officer was made for analysis purposes and data is exchanged based on the query, then it is obliged for the CIE officer to use that data for analysis purposes only:

```
[
obliged(analysisPurposesOnly(R, DATA_UNIT_ID))
<-
holds(analysisQuery(R, QUERY)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rAnalysisQueryAP(S, R, QUERY, DATA_UNIT_ID)).

requested(exchangeData(o, p, adriana,2)).
holds(analysisQuery(o, adriana)).
valid(rAnalysisQueryAP(o, p, adriana, 2)).
]
```
$\vdash_{justified}$ obliged(analysisPurposesOnly(p, 2))

Therefore, the requesting agent p accepts the offer and makes the following move.

p_3: accept(p, o, exchangeData(o, p, adriana, 2), holds(analysisPurposesOnly(p, 2)))

Finally the responding agent sends the requested intelligence data about Adriana.

5.2 Persuasion policies

This section presents the policies for persuasion. First the design choices are discussed. Then the persuasion policies are specified.

5.2.1 Design choices

Assumptions like cooperativeness and protection of own data are "hard coded" into the policies since they are applicable in all domains where data is exchanged between organizations. For example, an agent is cooperative in that he only asks *why P* questions if he does not have an argument for *P*. A second example of cooperativeness is that an agent only moves arguments in dialogues which are not overruled by his own arguments. So even though the opponent may not have a stronger counterargument to a certain argument, the agent does not move the argument if it is overruled by one of his own arguments. In the context of regulated data exchange between CIEs, cooperativeness promotes the goal to execute the police task by enabling more data exchange and data protection promotes the goal to protect local investigations and informants.

The persuasion policies are partly domain-specific in the parameterization of the required status types of arguments and the evaluation of arguments an agent needs for his decisions. Therefore, when applying the policies in an application domain, the required status types of arguments (justified or defensible) and the evaluation of arguments need to be specified. Specifying the status types of arguments can have a significant impact on the behavior of an agent and can be used to incorporate the attitude of CIEs towards data exchange (see section 2.6 on page 37). For example, a cooperative agent who accepts defensible arguments that say that it is not forbidden to exchange data will be easier persuaded than an overcautious agent who only accepts justified arguments for the conclusion that giving the data is not forbidden. So the cooperative agent, who needs a defensible argument, will not be as protective of his intelligence data as an overcautious agent, who needs a justified argument. The evaluation of arguments is also partly domain-specific. For the evaluation of arguments, domain-specific *context criteria* are used to determine whether arguments can be accepted, challenged or preferred to other arguments (see section 5.3 on page 111). For instance, domain-specific elements are that arguments are always accepted when a person with a particular role states them or that domain rules based on legislation are preferred to local rules or that factual statements which are considered reliable are never challenged.

Top-level arguments

Let us first look how an agent can respond to a *why P* move in a persuasion dialogue and determine how detailed the response should be. Consider the following start of a persuasion dialogue:

```
p1:     claim c(2)

o2:     why c(2)
```

What is at step p3 the optimal move for agent p to promote his goals? To further the goal to execute the appointed police task by cooperating, p should give the argument as reaction to the 'why c(2)' move. However, to further the goal to protect local investigations and informants, p must give no or minimal intelligence data. Now assume that the complete argument for 'c(2)' p can construct on the basis of his knowledge base is 'c(2) since c(X) <- b(X). b(X) <- a(X). a(2).'. The balance of goals is found by giving the minimal version of the argument (the *top-level argument*), which in this case is 'c(2) since c(X) <- b(X). b(2).'. The top-level argument promotes the goal to execute the appointed police task by cooperating by giving an argument, while furthering the goal to protect local investigations and informants by giving the minimal version of the argument. Therefore, at step p3 agent p makes the following move:

```
p3:     c(2) since c(X) <- b(X). b(2).
```

Definition 12 of Amgoud et al. (2006) does not explicitly define how to construct top-level arguments. Therefore, the persuasion policy must be able to construct top-level arguments from arguments. Generally, a top-level argument can be defined as:

Given an argument A with conclusion c, the top-level argument A_{top} is the minimal subargument of A with conclusion c. Using the ASPIC definition of an argument ASPIC, the top-level can now be defined.

Definition 9 Top-level arguments

Given argument P since $A_1 \ldots A_n$, its top-level argument A_{top} is:

P since CONC(A_1)…CONC(A_n).

Challenging factual statements

Let us comment on the design choices on how to challenge factual statements. Consider the following start of a persuasion dialogue:

```
p1:     claim c(2)

o2:     why c(2)

p3:     c(2) since c(X) <- b(X). b(2).
```

If '`b(2)`' is derivable (with sufficient force) from o's knowledge base, then o should concede '`b(2)`' to further the goal to execute the appointed police task. However, assume instead that nothing follows from o's knowledge base about '`b(2)`'. As described above, the idea is that o can apply context criteria to check whether '`b(2)`' can be conceded. Assume that this check succeeds, so that o could concede '`b(2)`'. At first sight, it would seem that the goal to execute the appointed police task requires that o concedes '`b(2)`'. However, upon closer inspection it turns out that there are good reasons why o should instead challenge '`b(2)`'. The reason is that it might be that `p`'s argument '`c(2) since c(X) <- b(X). b(2).`' is just a top-level argument and that `p`'s premise '`b(2)`' is based on an other argument, say, '`b(2) since b(X) <- a(X). a(2).`'. Now, it might be that o has a strong counterargument against this argument, or that o has strong reasons to believe '¬`a(2)`'. By challenging '`b(2)`', o can check whether this is indeed the case.

o_4: `why b(2)`

Now assume that the factual statement '`b(2)`' was the complete argument `p` can create, and that he cannot construct a more detailed argument for '`b(2)`'. Then `p` has to reply with the trivial argument:

p_5: `b(2) since b(2)`

At this stage, o knows that `p` has no further grounds for '`b(2)`', because the argument is of the form '*P since P*'. If o replied at step o_6 with a '`why b(2)`'move, then `p` would repeat his '`b(2) since b(2)`' reply which causes an infinite dialogue loop. Therefore, in situations where an argument is of the form '*P since P*' agents use their context criteria to determine whether the premise of the argument can be accepted. Assuming that o can accept the premises based on his context criteria, o concedes '`b(2)`'.

o_6: `concede b(2)`

Otherwise if o cannot accept the premises on his context criteria, o denies '`b(2)`'.

o_6: `deny b(2)`

This example also illustrates why an extra speech act *deny P* is added to the communication language of Van Veenen and Prakken (2006).

5.2.2 Specification of the persuasion policies

Persuasion policies determine how an agent should respond to (A) arguments, (B) *why* moves and (C) claims. The persuasion policies assume that an agent reasons with his internal knowledge base, which can be modified as a result of *concede*

moves. Recall further that the policies assume a selection mechanism to choose the moves to be replied to in a dialogue, where each chosen move serves as input for the policies. For each applicable policy step, the policy stores the reply move in *OutgoingMoves*. When every applicable policy step is taken, the policy returns all the reply moves contained in *OutgoingMoves*.

A: Reacting to an argument

The general idea of responding to arguments is to start with checking the context criterion on role authority. If, based on this criterion, the argument must be accepted then the premises and the conclusion of the argument are conceded. Otherwise, the second step is to determine whether the conclusion can be derived from the knowledge base. If this is possible then the conclusion of the argument is conceded. Otherwise, the third step is to check whether a counterargument can be derived from the knowledge base. If this is possible, a top-level argument for the negated conclusion is returned. Finally, the premises of the argument are evaluated.

1.
IF *IncomingMove* = argue(*S*, *R*, *Arg*)
THEN

If the agent can derive from the context criterion on the role of the other agent that the argument must be accepted then he concedes the premises and the conclusion of the argument. Recall that the satisfiesContextCriteria function now only check *S* (which denotes the sending agent) to determine the role of the agent.

1.1.
IF satisfiesContextCriteria(*S*)
THEN
FOREACH *P* in prem(*Arg*)
OutgoingMoves <- *OutgoingMoves* + concede(*R*, *S*, *P*)

OutgoingMoves <- *OutgoingMoves* + concede(*R*, *S*, conc(*Arg*))

Else, if the agent can derive the conclusion of the argument from his knowledge base, he concedes the conclusion.

1.2.
ELSEIF $[KB_{argumentation\text{-}engine}] \vdash_{status\text{-}type}$ conc(*Arg*)
THEN
OutgoingMoves <- *OutgoingMoves* + concede(*R*, *S*, conc(*Arg*))

Else, if the agent can create an undercut from his knowledge base, he returns an undercut for the argument. The `domainrule` function uses the `valid(R)` premise of the given argument to return the domain rule *R* used to construct the argument.

Note that only top-level arguments are created in the policy (see page 97), therefore arguments contain at most one domain rule and valid premise.

1.3.
ELSEIF

$[KB_{argumentation\text{-}engine}] \vdash_{status\text{-}type}$ ~domainrule(*Arg*)

THEN
OutgoingMoves <- *OutgoingMoves* + argue(*R*, *S*, *Arg'*)

Where argument *Arg'* is a top-level argument with the required status (defensible or justified) which has ~domainrule(*Arg*) as its conclusion.

Else, if the agent can derive the negated conclusion for the argument from his knowledge base, he returns a top-level argument for the negated conclusion (see the discussion about top-level arguments in subsection 5.2.2 on page 97).

1.4.
ELSEIF $[KB_{argumentation\text{-}engine}] \vdash_{status\text{-}type}$ ~conc(*Arg*)
THEN
OutgoingMoves <- *OutgoingMoves* + argue(*R*, *S*, *Arg'*)

Where argument *Arg'* is a top-level argument with the required status (defensible or justified) which has ~conc(*Arg*) as its conclusion.

Persuasion policy step 1.5 evaluates the premises of the argument. Note that the context criterion on role authority is not used in the evaluation of the premises since this is already applied in previous steps of the policy. Note also that the premises of an argument are always checked.

1.5.
FOR-EACH *P* in prem(*Arg*)

If premise *P* is the same as the conclusion of the argument (i.e., the argument is of the form *P since P*), then apply the remaining context criteria to determine whether *p* can be accepted. See also ‘Challenging factual statements’ on page 98).

```
1.5.1.
     IF conclusion(Arg) = P
     THEN
```

If the agent can derive from the context data that premise *P* must be accepted, then he concedes *P*. Note that the `satisfiesContextCriteria` function now only checks *P* (denoting the premise). Note further that in this step also the agent’s knowledge base will be updated (see subsection 5.2.3 on page 106 for the discussion about this).

```
1.5.1.1.
           IF satisfiesContextCriteria(P)
           THEN
                  OutgoingMoves <- OutgoingMoves +
                  concede(R, S, P)

                  updateKnowledgeBase(R, P)
```

else the agent denies *P*. Policy step 1.5.1.2 states that premise *P* is denied if the argument is of the form *P since P* and it is not possible to concede *P* based on the applied context criteria, see also Challenging factual statements on page 97).

```
1.5.1.2.
           ELSE
                  OutgoingMoves <- OutgoingMoves +
                  deny(R, S, P)
```

If the agent can derive premise *P* from his knowledge base, he concedes *P*,

```
1.5.2.
     ELSEIF [KB argumentation-engine] ⊢ status-type P
     THEN
           OutgoingMoves <- OutgoingMoves + concede(R, S, P)
```

else if the agent can derive ~*P* (the negation of *P*) from the knowledge base he returns the top-level argument for ~*P*, see 'Top-level arguments' on page 97.

1.5.3.

ELSEIF $[KB_{argumentation\text{-}engine}] \vdash_{status\text{-}type} \sim P$
THEN
OutgoingMoves <- *OutgoingMoves* + argue(*R*, *S*, *Arg'*)

Where argument *Arg'* is a top-level argument with the required status (defensible or justified) which has ~*P* as its conclusion.

Policy step 1.5.4 states that if *P* is a premise and cannot be derived from the agent's knowledge base and of the argument is not of the form *P since P*, then the agent asks why *P*. In policy step 1.5.4 an agent is cooperative in that he only asks *why P* questions if he does not have an argument for *P*.

1.5.4.
ELSE
OutgoingMoves <- *OutgoingMoves* + why(*R*, *S*, *P*)

B: Reacting to a why

The general idea for responding to *why P* moves is, when possible, to give an argument for *P*.

2.
ELSEIF *IncomingMove* = why(*S*, *R*, *P*)
THEN

At this stage in the policy, it is possible that the agent can no longer derive *P* from his knowledge base as a result from updating his knowledge during the processing of other parts of the dialogue. If this is the case, the agent retracts *P*.

2.1.
IF $[KB_{argumentation\text{-}engine}] \nvdash_{status\text{-}type} P$
THEN
OutgoingMoves <- *OutgoingMoves* + retract(*R*, *S*, *P*)

Else, the agent returns a top-level argument for *P*.

```
2.2.
     ELSE
            OutgoingMoves <- OutgoingMoves + argue(R, S, Arg')

            Where argument Arg' is a top-level argument with the
            required status (defensible or justified) which has P
            as its conclusion.
```

C: Reacting to a claim

The idea of responding to a claim is to start with checking the role authority criterion. If, based on this criterion, the agent concludes that the claim must be accepted then he concedes the claim. The second step is to determine whether the claim can be derived from the knowledge base. If this is possible then the claim is conceded. The third step is to check whether a counterargument for the claim can be derived from the knowledge base. If this is possible, a top-level argument for the negation of the claim is returned. If this is not possible a *why* move is returned

```
3.
    ELSEIF IncomingMove = claim(S, R, P)
```

If the agent can derive from the context criterion on the role of the other agent that the claim must be accepted, then he concedes the claim. Recall that the `satisfiesContextCriteria` function now only checks `S` (denoting the sending agent) to determine the role of the agent role.

```
3.1.
     IF     satisfiesContextCriteria(S)
     THEN
            OutgoingMoves <- OutgoingMoves + concede(R, S, P)
```

Else, if the agent can derive the claim from his knowledge base, he concedes the claim.

```
3.2.
     ELSEIF [KB argumentation-engine] ⊢ status-type P
     THEN
            OutgoingMoves <- OutgoingMoves + concede(R, S, P)
```

Else, if the agent can derive the negation of the claim from his knowledge base, he returns a top-level argument for the negated claim

3.3.

ELSEIF $[KB_{argumentation\text{-}engine}] \vdash_{status\text{-}type} \sim P$
THEN
OutgoingMoves <- *OutgoingMoves* + argue(*R*, *S*, *Arg'*)

Where argument *Arg'* is a top-level argument with the required status (defensible or justified) which has ~*P* as its conclusion.

Otherwise, ask why

3.4.
ELSE
OutgoingMoves <- *OutgoingMoves* + why(*R*, *S*, *P*)

Finally, in step 4, the persuasion policy returns the resulting move(s) from the previous steps to the agent calling the policy

4.

RETURN *OutgoingMoves*

The persuasion policy steps can be summarized as follows.

1. If argument *Arg*, then
   1.1. if *Arg* satisfies context criterion about role authority, then concede premises and conclusion;
   1.2. else if KB implies conclusion *Arg*, then concede conclusion *Arg*;
   1.3. else if KB implies undercut for *Arg*, then state undercut argument for *Arg;*
   1.4. else if KB warrants counterargument for *Arg*, then state counterargument for *Arg*;
   1.5. check each premise *P* of *Arg*
      1.5.1. if *Arg* is of the form *P since P*, then
         1.5.1.1. if *P* satisfies other context criteria (factual statements and domain rules), then concede *P*;
         1.5.1.2. else deny *P*;

1.5.2. if KB implies *P*, then concede *P*;

1.5.3. if KB warrants argument for ~*P*, then state argument for ~*P*;

1.5.4. else *why P*;

2. else if *why P*, then

   2.1. if KB does not imply conclusion *P*, then retract *P*;

   2.2. else give argument for *P*;

3. else if *claim P*, then

   3.1. if satisfies context criterion about role authority, then concede *P*;

   3.2. else if KB implies *P*, then concede *P*;

   3.3. if KB implies ~*P*, then state argument for ~*P*;

   3.4. else ask why *P*.

5.2.3 Knowledge updates

One remaining issue in the specification of the persuasion policies is the determination in which cases the agent's internal knowledge base must be updated because of concede or retract moves. Furthermore, as a result of knowledge base updates, it is possible that previous concede moves can no longer be maintained at later stages in dialogues. For that reason, also the cases in which (intended) concede moves need to be reevaluated must be determined.

Let us first investigate if knowledge must be updated as a result of concede moves. The first case in which an agent's knowledge possibly needs an update as result of making concede moves is: if the conclusion or premise of an argument is conceded because it is derivable from the knowledge base, should they also be added to the agent's knowledge base? In this case, updating the knowledge base is not required since the conclusion or premise can already be derived from the knowledge base.

The second case in which an agent's knowledge base possibly needs an update as result of making concede moves is: if both the premises and the conclusion of an argument are conceded based on the role of the other agent, should they be added to the agent's knowledge base? If the conclusion of an argument is added to the local knowledge base, then the conclusion becomes a factual statement in the knowledge base, which is not desired, since a conclusion has a different status than a factual

statement. Another option is to concede the conclusion but not add it to the knowledge base. This option is more desirable, since it allows for conceding moves based on agent role, while not changing the status of the conclusions. Moreover, when a conclusion is conceded based on agent role, while a justified argument against the conclusion can be created based on the knowledge base, then a choice has to be made which premise must be removed from the knowledge base to be able to create a justified argument for the conclusion. Choosing which premises must be removed from the knowledge base is a problem of belief revision (see Gärdenfors 1992), which is beyond the scope of this thesis and is a subject for further research. Similar to conceding a conclusion based on agent role, it is more desirable that premises will not be added to the knowledge base. The policies of this research allow conceding both the premises and the conclusion of an argument based on satisfying the context criterion on agent role (see persuasion policy step 1.1 on page 100 and step 3.1 on page 104). Since it is more desirable to not update the knowledge base if both the premises and the conclusion of an argument are conceded based on agent role, the policies of this research implement this preference. Recall that by using context criteria, the policies are made partly domain-specific. An example of satisfying a context criterion on agent role is that the other agent is trusted. In the Dutch police domain, both the premises and the conclusion of an argument are conceded if the other agent has a high authority (see also section 5.3 on context criteria on page 111).

The third case in which an agent's knowledge possibly needs an update as result of making concede moves is: if premises of an argument are conceded based on other criteria, should they be added to the agent's knowledge base? The policies of the present research also allow conceding premises of an argument based on other criteria. In the Dutch police domain, example criteria for accepting premises are: whether the origin of the domain rule is the law or whether the supplier of the factual statement is reliable (see also section 5.3 on page 111). Adding those premises does not change their status, since premises which are conceded based on other context criteria are unknown domain rules or factual statements. Therefore, in the policies of this research, premises of an argument are added to the knowledge base if they are conceded based satisfying other criteria (see persuasion policy step 1.5.1.1 on page 102). Note that the policies of the present research use the following method to add unknown domain rules to a knowledge base: when an agent concedes a valid(r) premise, he also adds domain rule r to his knowledge base.

Let us next investigate if knowledge must be updated as a result of retract moves. The first case in which an agent's knowledge possibly needs an update as result of making retract moves is: if a conclusion is retracted, should premises then be removed from the knowledge base? Removing premises from the knowledge base as a result of retracting a conclusion will also lead to belief revision problems.

However, this situation does not occur in the persuasion policies of the present research, since only premises or claims can be retracted as a possible reaction to *why* moves. The *why* moves can be observed in persuasion policy step 1.5.4 (page 103) and in persuasion policy step 3.4 (page 105).

The second case in which an agent's knowledge possibly needs an update as result of making retract moves is: if a premise is retracted, should things be then be removed from the knowledge base? Similar to retracting conclusions, this will lead to belief revision problems. Fortunately, in the policies of the present research, retract moves are only made when an agent cannot derive a premise or claim as a result from previous knowledge base updates. This can be observed in persuasion policy step 2.1 (page 103), which is the only case when the policy prescribes a retract move. Therefore, the knowledge base does not need to be updated and belief revision problems do not occur.

One complication is that during dialogues, previous concede moves can be undermined by knowledge updates at later stages. To illustrate this complication, let us assume that a knowledge base of `o` consists of the following factual statement and rules:

```
{c(X) <- a(X) 0.1. ~c(X) <- b(X) 0.9. a(2).}
```

and consider the following start of a persuasion dialogue between `p` and `o`:

```
p1:     d(2) since d(X) <- c(X), b(X). c(2). b(2).
```

The first premise of the argument is '`c(2)`', which is derivable from `o`'s knowledge base. Therefore, based on persuasion policy step 1.5.2 (page 102), agent `o` intends to concede to the first premise in move o_2 and puts this move in his outgoing moves:

```
OutgoingMoves = {concede c(2)}
```

For the second premise of the argument is '`b(2)`', persuasion policy step 1.5.4 (page 103) is applicable. Based on step 1.5.4, which states that if '`b(2)`' is a premise and cannot be derived from the agent's knowledge base and of the argument is not of the form *`P since P`*, he adds '`why b(2)`' to his outgoing moves:

```
OutgoingMoves = {concede c(2), why b(2)}
```

Agent `o` has processed the incoming move and states the outgoing moves at his turn in the dialogue:

```
o2:     concede c(2)
```

o_3: why b(2)

At this turn, agent p chooses to react to move o_3 with 'b(2) since b(2)' and makes the following move:

p_4: b(2) since b(2)

Assuming that b(2) satisfies agent o's context criteria, persuasion policy step 1.5.1.1 (page 102) is applicable. Thus o puts 'concede b(2)' in his outgoing moves.

OutgoingMoves = {concede b(2)}

At this moment o's knowledge base is updated with the addition of factual statement 'b(2)'. As a result, o's knowledge base now consists of the following factual statements and rules:

KB_o = {c(X) <- a(X) 0.1. ~c(X) <- b(X) 0.9. a(2). b(2).}.

However, at this stage o can no longer derive 'c(2)' because domain rule '~c(X) <- b(X)' is stronger than 'c(X) <- a(X)'. Thus move o_2 where c(2) was conceded can no longer be maintained. After updating his knowledge base, agent o makes the following move:

o_5: concede b(2)

This example illustrates that it is possible that previous concede moves no longer can be maintained at later stages in dialogues. The concession made at dialogue step o_2 could no longer be maintained after knowledge update after dialogue step o_4. A pragmatic solution is to regard concedes at previous turns as definitive. This is in fact modeled by protocol rule R_5 from chapter 4 (on page 65) which states that once a move is surrendered, it may not be revoked.

Intended concede moves can also be undermined by knowledge updates within a turn. To illustrate this, let us start with the same dialogue as in the previous example:

KB_o = {c(X) <- a(X) 0.1. ~c(X) <- b(X) 0.9. a(2).}

p_1: d(2) since d(X) <- c(X), b(X). c(2). b(2).

The first premise of the argument 'c(2)' is derivable from o's knowledge base. So, at this turn o intends to concede to the first premise and agent o puts this move in his outgoing moves:

OutgoingMoves = {concede c(2)}

Assume next that agent o intends to concede to the second premise 'b(2)' of the argument based on reliability. In the persuasion policies of the present research, this move is not allowed because premises of arguments are only accepted if they have the form *P* since *P*. However, a persuasion policy with a different design could accept premises of an argument based on context criteria. Since agent o intends to concede to the second premise 'b(2)', he adds 'concede b(2)' to his outgoing moves:

OutgoingMoves = {concede c(2), concede b(2)}

At this moment o's knowledge base is updated with the addition of factual statement 'b(2)'. As a result, o's knowledge base now consists of the following factual statements and rules:

KB_o = {c(X) <- a(X) 0.1. ~c(X) <- b(X) 0.9. a(2). b(2).}.

However, at this stage o can no longer derive 'c(2)' since domain rule '~c(X) <- b(X)' is stronger than 'c(X) <- a(X)'.

A practical way to solve this knowledge update complication is for every turn to reassess the intended concede moves before putting them in the dialogue. A way to reassess the intended moves is to do a knowledge update after every concession based on context criteria. When an intended concede is undermined by the knowledge update(s), the intended concede move is retracted from the outgoing moves. With this pragmatic solution, agent o checks his outgoing moves {concede c(2), concede b(2)}. The first intended concede move 'concede c(2)' is no longer possible. Therefore, this move is retracted from the outgoing moves. The second intended move 'concede b(2)' is still possible and remains in o's outgoing moves. Thus with the pragmatic solution, the dialogue continues with move o_2 as:

o_2: concede b(2)

To the best of my knowledge the complication that concede moves can be undermined by knowledge updates in a dialogue is a new problem. For the present research I have chosen for a pragmatic solution. However, in future research this identified knowledge update problem needs to be investigated more systematically.

5.3 Application of dialogue policies in the Dutch police domain

This section discusses the application of the policies in an application domain. First the dialogue policies are made domain-specific, i.e., the evaluation of arguments particular for the Dutch police domain is specified. Then an example is presented to illustrate the policies for the Dutch police domain.

Context criteria

For the evaluation of arguments, domain-specific *context criteria* are used to determine whether arguments can be accepted, challenged or preferred to other arguments. In the Dutch police domain, three categories of context criteria can be discerned. The first context criteria category is related to agents, the second category is associated to domain rules and the third category is related to factual statements.

Context criterion about agents: role authority

The context criterion on role authority indicates the authority derived from the role that a person fulfils in an organization. For example, if a software agent has the role of someone who has a high authority in the organization then arguments from that agent are more easily accepted.

Context criterion about domain rules: rule source

The context criterion about rule source indicates the source of a domain rule. For instance, if the source of a domain rule is the law, then this rule will be more easily accepted than local interpretations.

Context criterion about factual statements: data reliability

The context criterion about data reliability indicates the trustworthiness of factual statements. For example, an argument based on an unreliable factual statement will not easily be accepted. As described in section 2.2 (page 13), in the police domain data reliability is split up into two parts: the reliability of the supplier of the data and the reliability of the data itself. For instance, if an informant was trustworthy in the past, then data originating from him will have more value and therefore the data supplier reliability will be evaluated as trustworthy, and if the supplied data is confirmed by an police officer or police report then the data itself are considered to be reliable.

For the specification of the context criteria for the Dutch police domain, two algorithms are required. The first algorithm, which implements the context criterion on role authority, has the sending agent as parameter (denoted by s). The first

algorithm states that if the other agent has a high role authority, then the `SatisfiesContextCriteria(S)` returns true, in all other cases the algorithm returns false. When the `SatisfiesContextCriteria(S)` algorithm returns true, the agent concludes that the argument or claim from the other agent must be conceded.

```
SatisfiesContextCriteria(S)

IF roleAuthority(S) = high
THEN
        RETURN true
ELSE
        RETURN false
```

The second algorithm, which implements the context criteria on domain rules and factual statements, has a premise as parameter (denoted by `P`). The second algorithm distinguishes between premises based on inference rules to apply the context criterion on rule source, and factual statements to apply the context criterion on factual statements.

When the premise is an inference rule and the source of the rule is the law or a local interpretation of the law, then the `SatisfiesContextCriteria(P)` algorithm returns true, otherwise it returns false. When the premise is a factual statement and the reliability of the data is considered as true and the data supplier is considered to be reliable, then the `SatisfiesContextCriteria(P)` algorithm returns true. Note that data reliability value `1` (specified in the algorithm with `reliabilityData(P) = 1`) denotes that the data is considered as true, for instance because the data is verified by a police officer. Furthermore, `reliabilityDataSupplier(P) = a` denotes that the data supplier is reliable (see section 2.2 on page 13).

```
SatisfiesContextCriteria(P)

IF inferenceRule(P) = true
THEN
        IF ruleSource(P) = law OR ruleSource(P) = localInterpretation
        THEN
                RETURN true
        ELSE
                RETURN false

ELSE //P is a factual statement
        IF reliabilityData(P) = 1 AND reliabilityDataSupplier(P) = a
        THEN
                RETURN true
        ELSE
                RETURN false
```

Example 8 terrorist assault on the queen

Let us illustrate a combination of negotiation and persuasion dialogue policies with an example from the Dutch police domain. In this example (which is based on Example 2), the requesting agent `p` requests the responding agent `o` to offer him intelligence data about a terrorist group in Amsterdam which is possibly planning an assault on the queen. Initially the responding agent rejects his request. However, the requesting agent persuades him to exchange the requested data.

The knowledge bases of both agents contain the following domain rules (see appendix section starting on page 159): *rOlderThanOneYear_F, rExceptionalSituation_OTOY, rQueen_ES*, *rWpolr14a_1*. The responding agent has '`holds( dataOlderThanOneYear(3))`' as factual statement in his knowledge base. The requesting agent has '`holds(attemptAssaultQueen(3))`' as factual statement is his knowledge base. Furthermore, both agents have a low role authority, apply the context criteria specified in on page 111, and their dialogue policies use justified arguments in the selection of allowed moves. Note that Figure 1 gives a graphical representation of the reply structure of this dialogue. The dialogue starts with the request from agent `p` to agent `o` to give him intelligence data about a possible terrorist assault:

p_1: `request(p, o, exchangeData(o, p, terrorists, DATA_UNIT_ID),)`

The responding agent `o` finds a matching data-unit identifier ('`3`') matching the query. However, he rejects the request because negotiation policy step 2.3 is applicable:

```
IF [KB argumentation-engine ∪
{requested(exchangeData(o, p, terrorists, 3))}]
⊢justified
obliged(not(exchangeData(o, p, terrorists, 3)))
THEN
        OutgoingMoves <- OutgoingMoves +
        reject(o, p, exchangeData(o, p, terrorists, 3),)
```

The rejection in negotiation policy step 2.3 is based upon the domain rule *rOlderThanOneYear_F* which states that it is forbidden to exchange requested data older than one year (see page 168) and that the found data unit '`3`' is older than one year:

```
[
obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID))
<-
holds(dataOlderThanOneYear(DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)) 0.6.
```

```
holds(dataOlderThanOneYear(3)).
valid(rOlderThanOneYear_F(o, p, terrorists, 3)) 0.6.
]
⊢justified obliged(not(exchangeData(o, p, terrorists, 3)))
```

Therefore, the responding agent `o` rejects the request to give agent `p` the found intelligence data about terrorist attack:

o_2: `reject(o, p, exchangeData(o, p, terrorists, 3),)`

Since the incoming move is a *reject*, agent `p` applies step 3.1 of the negotiation policy.

Policy step 3.1 first backtracks to determine if the *reject* move has not been made before. This is the case, and therefore he reacts with a *why-reject* move that will initiate an embedded persuasion dialogue. Until it is terminated, no negotiation moves are allowed by the protocol:

p_3: `why-reject(p, o, exchangeData(o, p, terrorists, 3))`

In responding to the *why-reject* move, agent `o` first applies negotiation policy step 4.1. In this step the negotiation policy checks if the argumentation system can create a justified argument for the fact it is forbidden to exchange the data:

```
IF [KB_argumentation-engine ∪ {requested(exchangeData(o, p,
terrorists, 3))}]
⊢justified obliged(not(exchangeData(o, p, terrorists, 3)))
THEN
        OutgoingMoves <- OutgoingMoves +
        claim(R, S, obliged(not(Arg)))
```

This is possible, based upon domain rule *rOlderThanOneYear_F*:

```
[
obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID))
<-
holds(dataOlderThanOneYear(DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)) 0.6.

holds(dataOlderThanOneYear(3)).
requested(exchangeData(o, p, terrorists, 3)).
valid(rOlderThanOneYear_F(o, p, terrorists, 3)) 0.6.
]
⊢justified obliged(not(exchangeData(o, p, terrorists, 3)))
```

Therefore the conclusion that it is forbidden to exchange data about terrorist activities is returned as the claim.

o_4: `claim(o, p, obliged(not(exchangeData(o, p, terrorists, 3)))`

Now agent p applies step 3 of the persuasion policy. Agent `p` cannot concede the claim based on his context criteria (persuasion policy step 3.1), since agent `o` has a low role authority. If the responding agent `o` had a high role authority, agent p had to concede the claim and therefore could not persuade the responding agent to exchange the requested data. Moreover, requesting agent `p` cannot derive the claim from his knowledge base (step 3.2) and also cannot create a counterargument (step 3.3), so he applies step 3.4 to ask 'why is it forbidden to exchange that data?'.

p_5: `why(p, o, obliged(not(exchangeData((o, p, terrorists, 3))))`

To react to the *why* move, agent `o` applies step 2.1 of the persuasion policy. In step 2.1, the policy first checks if the argumentation system cannot derive the claim (which can occur as a result of updating his knowledge during the processing of other parts of the dialogue); it is possible to derive the claim, therefore step 2.2 is applicable. In step 2.2 a top-level argument (which in this case equals the argument) 'it is forbidden to exchange the data since the data is too old' for the claim is returned.

o_6:
```
argue(o, p,
obliged(not(exchangeData(o, p, terrorists, 3))).

obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID))
<-
holds(dataOlderThanOneYear(DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)).

holds(dataOlderThanOneYear(3)).
requested(exchangeData(o, p, terrorists, 3)).
valid(rOlderThanOneYear_F(o, p, terrorists, 3))).
)
```

To respond to the argument, agent p applies step 1 of the persuasion policy. In step 1.1 he checks if the argument can be accepted on the basis of the context criteria, which is not the case (similar to the response from agent `p` to move o_4). Therefore, he continues with step 1.2 where he checks if he can derive the conclusion of the argument from his knowledge base, this also not the case. Step 1.3 is applicable, because he can create an undercut for the argument:

```
[
~rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)
<-
holds(exceptionalSituation(DATA_UNIT_ID)),
valid(rExceptionalSituation_OTOY(S, R, QUERY,
DATA_UNIT_ID)).

holds(exceptionalSituation(DATA_UNIT_ID))
<-
holds(attemptAssaultQueen(DATA_UNIT_ID)),
valid(rQueen_ES(DATA_UNIT_ID)).

valid(rExceptionalSituation_OTOY(o, p, terrorists, 3)).
holds(attemptAssaultQueen(3)).
valid(rQueen_ES(3)).
]
⊢justified ~rOlderThanOneYear_F(o, p, terrorist, 3).
```

The undercut argument for `~rOlderThanOneYear_F(o, p, terrorists, 3)` is based on the domain rules *rExceptionalSituation_OTOY* and *rQueen_ES*. The domain rule *rExceptionalSituation_OTOY* (see page 170) states that in a exceptional situation the rule stating that it forbidden to exchange requested data older than one year is not applicable, and the domain rule *rQueen_ES* (see page 169) states that if there is an attempted assault on the head of state then an exceptional situation occurs.

Based on domain rules *rExceptionalSituation_OTOY, rQueen_ES* and the factual statement '`holds(attemptAssaultQueen(3)).`' the agent concludes that domain rule *rOlderThanOneYear_F,* is not applicable in this situation and puts the top-level argument

```
argue(p, o,
~rOlderThanOneYear_F(o, p, terrorist, 3).

~rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)
<-
holds(exceptionalSituation(DATA_UNIT_ID)),
valid(rExceptionalSituation_OTOY(S,
R,QUERY,DATA_UNIT_ID)).

holds(exceptionalSituation(3)).
valid(rExceptionalSituation_OTOY(o, p, terrorists, 3))
)
```

in the *OutgoingMoves* (move p_7 reacting to move o_6).

Agent p continues with step 1.5 to evaluate the premises of the argument.

The first premise of the argument is '`holds(dataOlderThanOneYear(3))`'. Since the argument is not of the form *p since p* step 1.5.1 is no applicable. Step 1.5.2 and step 1.5.3 are not applicable since agent p cannot derive the premise or the negated premise of the argument. Therefore, step 1.5.4 is applicable, and he puts

```
why(p, o, holds(dataOlderThanOneYear(3)))
```

in his *OutgoingMoves*. (move p_8 reacting to move o_6)

The second premise of the argument is '`requested(exchangeData(o, p, terrorists, 3))`'.
Assuming that the requested action '`requested(exchangeData(o, p, terrorists, 3))`' is conjoined with the knowledge during the embedded persuasion dialogues, agent p can derive the second premise from his knowledge base. Therefore, persuasion policy step 1.5.2 is applicable, and he puts

```
concede(p, o, requested(exchangeData(o, p, terrorists, 3))
```

in his *OutgoingMoves* (move p_9 reacting to move o_6)

The third premise of the argument is '`valid(rOlderThanOneYear_F(o, p, terrorists, 3)`.' Since the argument is not of the form *p since p* step 1.5.1 is not applicable. However, he can create a justified argument which makes step 1.5.2 applicable, therefore he puts

```
concede(valid(rOlderThanOneYear_F(o, p, terrorists, 3)))
```

in the *OutgoingMoves* (move p_{10} reacting to move o_6).

Finally, in step 4 the policy returns the moves contained in *OutgoingMoves*, and the selection mechanism states moves at this turn in the dialogue. (moves p_7, p_8, p_9 and p_{10}).

p_7-o_6:
```
argue(p, o,
~rOlderThanOneYear_F(o, p, terrorist, 3).

~rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)
<-
holds(exceptionalSituation(DATA_UNIT_ID)),
valid(rExceptionalSituation_OTOY(S, R, QUERY, DATA_UNIT_ID)).
```

```
        holds(exceptionalSituation(3))
        valid(rExceptionalSituation_OTOY(o, p, terrorists, 3)).
        )
```

p_8-o_6: `why(p, o, holds(dataOlderThanOneYear(3)))`

p_9-o_6: `concede(p, o, requested(exchangeData(o, p, terrorists, 3))`

p_{10}-o_6:`concede(valid(rOlderThanOneYear_F(o, p, terrorists, 3)))`

At this turn the responding agent reacts to moves p_7 and p_8. The selection mechanism of the agent first passes move p_7 to the policy.

For move p_7, persuasion policy step 1.5 is applicable. In the evaluation the premises, steps 1.5.4 and 1.5.2 are applicable. Therefore, the responding agent makes the following moves.

o_{11}-p_7:`why(o, p, holds(exceptionalSituation(3)))`

o_{12}-p_7: `concede(o, p, valid(rExceptionalSituation_OTOY(o, p, terrorists, 3)))`

The selection mechanism of the agent then passes move p_8 to the policy. For move p_8, persuasion policy step 2.2 is applicable; therefore, the policy returns a top-level argument for `holds(dataOlderThanOneYear(3))`.

o_{13}-p_8:
```
argue(o, p,
        holds(dataOlderThanOneYear(3)).

        holds(dataOlderThanOneYear(3)).
        )
```

At this turn the requesting agent reacts to moves o_{11} and o_{13}. The selection mechanism of the agent first passes move o_{11} to the policy.

The selection mechanism of the agent then passes move o_{11} to the policy. For move o_{11}, persuasion policy step 2.2 is applicable. Therefore he puts the argument

```
argue(p, o,
holds(exceptionalSituation(3)).

holds(exceptionalSituation(DATA_UNIT_ID))
<-
holds(attemptAssaultQueen(DATA_UNIT_ID)),
valid(rQueen_ES(DATA_UNIT_ID)).
```

```
        holds(attemptAssaultQueen(3)).
        valid(rQueen_ES(3))
        )
```

in the *OutgoingMoves*.

For move o_{13}, persuasion policy step 1.5.1 is applicable, since the factual statement has a high reliability for both data and supplier and thus satisfying the context criteria, he puts

```
concede(p, o, holds(dataOlderThanOneYear(3)))
```

in his *OutgoingMoves* and adds the factual statement '`holds(dataOlderThanOneYear(3))`' to his knowledge base.

Finally, in step 4 the policy returns the moves contained in *OutgoingMoves*, and the selection mechanism states moves at this turn in the dialogue (move p_{14} as reaction to move o_{11} and move p_{15} as reaction to move o_{13}).

p_{14}-o_{11}:
```
argue(p, o,
        holds(exceptionalSituation(3)).

        holds(exceptionalSituation(DATA_UNIT_ID))
        <-
        holds(attemptAssaultQueen(DATA_UNIT_ID)),
        valid(rQueen_ES(DATA_UNIT_ID)).

        holds(attemptAssaultQueen(3)).
        valid(rQueen_ES(3)).
        )
```

p_{15}-o_{13}:
```
concede(p, o, holds(dataOlderThanOneYear(3)))
```

To react to the argument stated in move p_{14}, agent `o` applies step 1 of the persuasion policy. Steps 1.1, 1.2, 1.3 and 1.4 are not applicable and agent `o` continues with step 1.5 to evaluate the premises of the argument.

The first premise of the argument is '`holds(attemptAssaultQueen(3))`'. Since step 1.5.4 is applicable, he puts

```
why(o, p, holds(attemptAssaultQueen(3)))
```

in his *OutgoingMoves* (move o_{16} as reaction to move p_{14}).

The second premise of the argument is '`valid(rQueen_ES(3))`'. Since step 1.5.2 is applicable, he puts

```
concede(o, p, valid(rQueen_ES(3)))
```

in his *`OutgoingMoves`* (move o_{17} as reaction to move p_{14}).

Finally, in step 4 the policy returns the moves contained in *`OutgoingMoves`*, and the selection mechanism states those moves at this turn in the dialogue (move o_{16} and move o_{17} as reaction to move p_{14}):

o_{16}-p_{14}:`why(o, p, holds(attemptAssaultQueen(3)))`

o_{17}-p_{14}:`concede(o, p, valid(rQueen_ES(3)))`

At this turn the responding agent reacts to move o_{16} for which persuasion policy step 2.2 is applicable; therefore, the policy returns a top-level argument for `holds(attemptAssaultQueen(3))`.

p_{18}-o_{16}:`argue(p, o,`
`holds(attemptAssaultQueen(3)).`

`holds(attemptAssaultQueen(3)).`
`)`

Note that move p_{18} illustrates that the agents trust each other, and do not only negotiate and persuade.

Persuasion policy step 1.5.1.1 is applicable when reacting to move p_{18} since the argument is of the form *p since p* and satisfies his context criteria (similar to response p_{15} to move o_{13}). Therefore:

o_{19}-p_{18}:`concede(o, p, holds(attemptAssaultQueen(3)))`

Furthermore, the factual statement '`holds(attemptAssaultQueen(3))`' is added to his knowledge base.

Agent `o`'s original reason for '`obliged(not(exchangeData(o, p, terrorists, 3))`' is now defeated and therefore concedes to `p`'s negotiation. With this move the responding agent `o` backtracks to move p_5, now surrendering to that move (move o_{20} as reaction to move p_5). This move terminates the persuasion dialogue so that negotiation moves are allowed again by the protocol.

o_{20}-p_5: `retract(o, p, obliged(not(exchangeData(o, p, terrorists, 3)))`

After move o_{20} the agent backtracks to move p_1. Agent o now knows that it is not forbidden to exchange the requested data about terrorist attacks.
Domain rule *rWpolr14_a__1* states that it is obliged to exchange data with other CIE officers (based on the assumption that they need the data for the execution of the police task). Initially the argument based on domain rule *rWpolr14a__1* (see page 159) was rebutted by an argument based on rule *rOlderThanOneYear_F* (see move o_2), but is now reinstated. The argument based on rule *rOlderThanOneYear_F* is undercut by an argument based on the *rExceptionalSituation_OTOY* and *rQueen_ES* domain rules. Therefore, the responding agent offers to exchange data about terrorist attacks.

o_{21}-p_1: offer(o, p, exchangeData(o, p, terrorists, 3)))

The requesting agent accepts the offer.

p_{22}-o_{21}:accept(p, o, exchangeData(o, p, terrorists, 3)))

Finally, the responding agent sends the requested intelligence data about terrorist attacks.

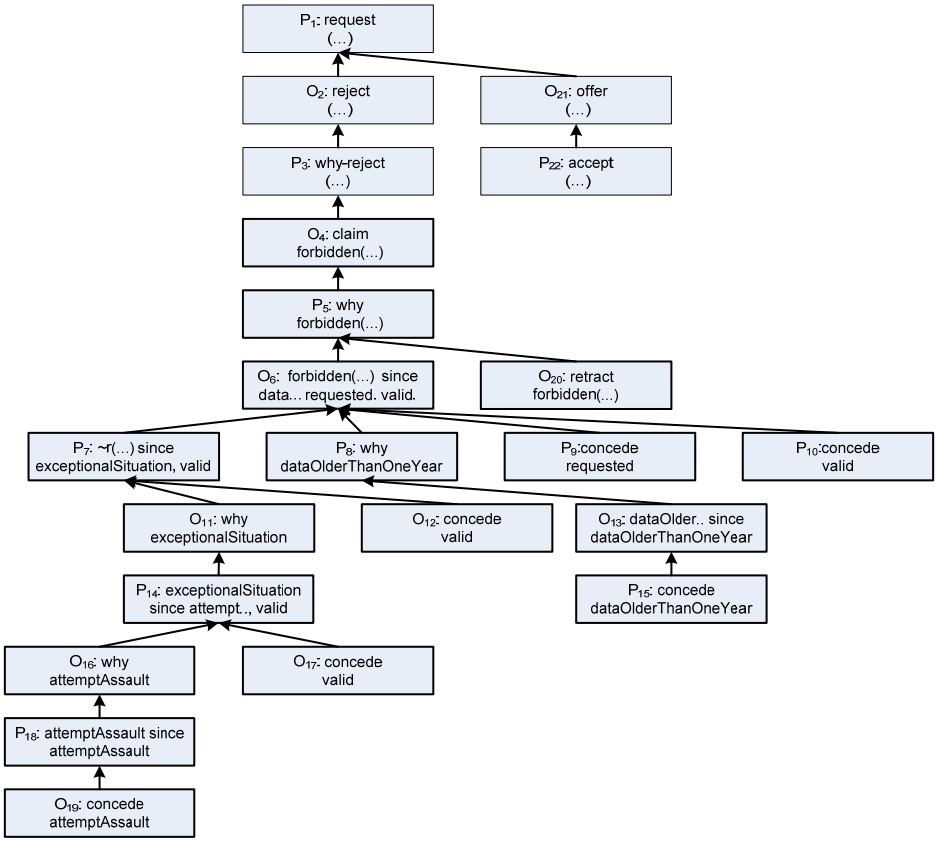

Figure 1: Reply structure of the terrorist attacks example

5.4 Chapter summary

This chapter presented the dialogue policies for negotiation with embedded persuasion. The policies promote the goal to execute the police task by enabling more data exchanges and the goal to protect local investigations and informants. For example, the negotiation policy for an offer could reject requests for data if there is no justified argument which says that it is obliged to exchange the found data. However, this action is not in line with the goal to execute the police task. Therefore the policy tries to exchange data such that it is not forbidden to exchange the requested data and that the agents' own interests are not violated. Also, when an agent receives a rejection, he will try to persuade the other agent to exchange intelligence data. A second example of how the policies promote the goal to execute the police task is that an agent only asks *why P* questions if he does not have an argument for *P*. Finally, an example of how the policies balance the goals is that they give the minimal version of arguments (*top-level argument*). Top-level arguments promote the goal to execute the appointed police task by cooperating by giving an argument (contrasted to not saying anything at all, or simply lying), while furthering the goal local investigations and informants by giving the minimal version of the argument.

Assumptions about cooperativeness and data protection are applicable in all domains where data is exchanged between organizations. Therefore those assumptions are "hard coded" into the policies. The dialogue policies are partly domain-specific in that the required status types (justified or defensible) of arguments and context criteria to evaluate arguments are parameterized. The next chapter presents a multi-agent system architecture where the dialogue policies are implemented to optimize regulated data exchange.

Chapter 6

Specification and implementation of the MAS architecture

This chapter describes the specification and implementation of the multi-agent system architecture for the Dutch police domain[19]. In section 6.1 the design of the multi-agent system and individual agent architecture is presented. In section 6.2 the implementation of the agent architecture is discussed.

6.1 MAS architecture design

This section discusses the design of the multi-agent system architecture. First the high-level architecture of the multi-agent system for data exchange between criminal investigation units is described (6.1.1). Then the architecture of an individual agent is specified (6.1.2).

6.1.1 Multi-agent system architecture

Figure 2 gives a high-level view of the multi-agent architecture consisting of a requesting and a responding agent. In this architecture, the requesting agent requests for data and the responding agent responds to requests from the requesting agent. Both agents have access to their data sources within their local organization and act on behalf of their users. In the Dutch police domain, typical users are CIE officers who want to gather or protect intelligence data about severe crime. The agents communicate with other agents through the agent communication channel. Given that the architecture consists of two agents, multi-agent issues such as how agents can locate each other and how to communicate with multiple agents do not occur in the present situation. An assumption for the multi-agent architecture is that agents trust each other (see section 2.4 on page 18). Therefore, security issues such as identification (how is an agent identified), authentication (verifying the identity

[19] Note that this chapter is based on earlier work published in Dijkstra et al. (2007) .

of an agent) and authorization (who is qualified to access a data source) are subject for further research.

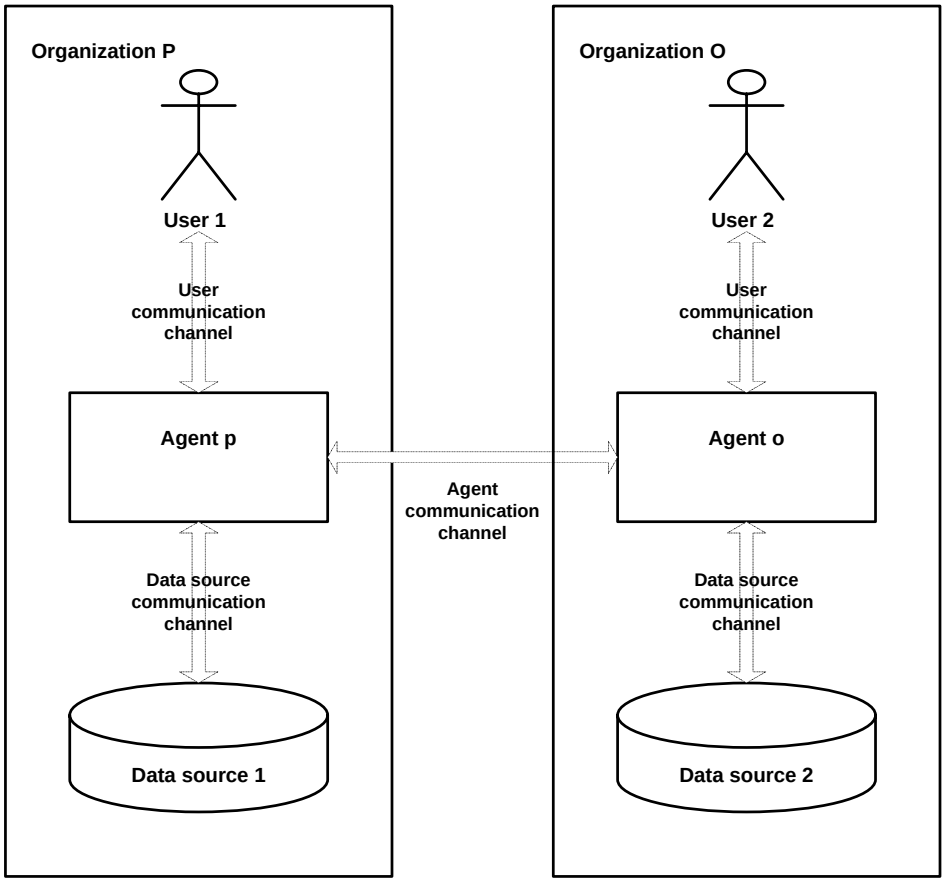

Figure 2: The multi-agent architecture

When user *1* from organization *P* (see Figure 2) wants to obtain data from organization *O*, he submits a search query to agent *p*. On behalf of user *1*, the requesting agent *p* will request data from agent *o*. The responding agent *o* will search in data source *2* for data matching the request. Depending on his knowledge, agent *o* can give the following responses to agent *p*: that that no matching are data found, that data is found which can be exchanged, that data is found which can only be exchanged under certain conditions, or that data is found which cannot be exchanged. In the present research, the focus is on scenarios where data is found but

where the responding agent infers that the found data can only be exchanged under certain conditions (as illustrated in Example 1: source protection & analysis purposes on page 33) and on scenarios where the responding agent infers that the found data cannot be exchanged (as illustrated in Example 3: the investigation protection & national importance on page 36).

As described in section 3.3 (page 47), dialogues between agents about regulated data exchange are interpreted as dialogues starting as a negotiation that may shift to embedded persuasion. The dialogue protocol suitable for the required interactions was specified and formalized in chapter 4. The policies for the negotiation and persuasion dialogues were specified and formalized in chapter 5. Recall that the key idea of the dialogue policies is that they prescribe the best reaction during the dialogues, i.e., the reaction which is expected to further the different goals while respecting the attitude of the criminal investigation unit towards data exchange.

To allow this research to be generalizable to other domains, such as to regulated data exchange in the medical domain or between tax authorities, terminology specific to the Dutch police is abstracted in the MAS architecture. In the Dutch police domain, data is divided into separate registrations, which in turn consist of mutations (see section 2.2 on page 13). Based upon the assumption (see section 2.4 on page 18) that data is stored in local severe-crime registers, registers will be abstracted as data sources and mutations as data units. A data unit is a grouped piece of data, which has an associated identifier. The data usage codes (see subsection 2.4.3 on page 25) are abstracted as metadata used to describe data. Furthermore, the criminal investigation units are abstracted as organizations. The agents have an ontology (Gruber 1993) of the domain to represent knowledge of the data sources, applicable regulations and to be able to exchange data. Ontology research is a broad research area, for example how to achieve interoperability between concepts belonging to heterogeneous ontologies, which is commonly called ontology alignment (Euzenat 2004). In this multi-agent system architecture, all agents use the same ontology of the domain so problems with different ontologies are avoided.

6.1.2 Individual agent architecture

This section describes the components of an individual agent (see Figure 3).

A: Communication modules

The user communication module provides for the external communication with the user of the agent. The external input is a query from the user and the external output is the outcome of the interaction between software agents. Internally, the user communication module parses the queries and the outcomes of the agent interactions between the execution cycle module and the user. For example, assume

that a CIE officer wants to gather intelligence data about Soprano from another criminal investigation unit. First the CIE officer enters 'Soprano' as a search query. Then the software agent will, on behalf of his CIE officer, search for intelligence data about Soprano. While searching for data, the software agent will interact with the software agent from the other criminal investigation unit. During the interaction, the software agent will sometimes negotiate and persuade to try to obtain all available data about Soprano. When interaction between the agents has ended, the agent will present the outcome of the interaction to his user.

The data source communication module is responsible for the communication with the external data source. The external data source contains data units and data-unit identifiers referring to those data units. A data unit is a grouped piece of data and has metadata to describe it.

The agent communication module provides for the communication with other agents and parses messages for further processing in the agent execution cycle. The data-unit identifiers are also used in the communication with other agents to refer anonymously to data units.

B: Execution cycle module

The execution cycle module processes messages from the agent, data and user communication modules, and from the move selector module. If an incoming message is a move originating from the agent communication module and the move is a *request*, then the move is first instantiated with a matching data-unit identifier (by calling the data source communication module). If the incoming message is an *accept* move, then the data unit corresponding to the data-unit identifier data is sent to the agent communication module by obtaining it from the data source communication module. Except for the *accept* and *withdraw* moves, the execution cycle module passes all incoming moves on to the move selector module. Finally, the outcome of a dialogue is passed on to the user communication module, which presents the outcome to the user of the requesting agent.

C: Move selector module

The move selector module processes the incoming moves from the execution cycle module and chooses the moves which will be replied to in a dialogue. The chosen moves, in turn, are passed on to the dialogue policies module.

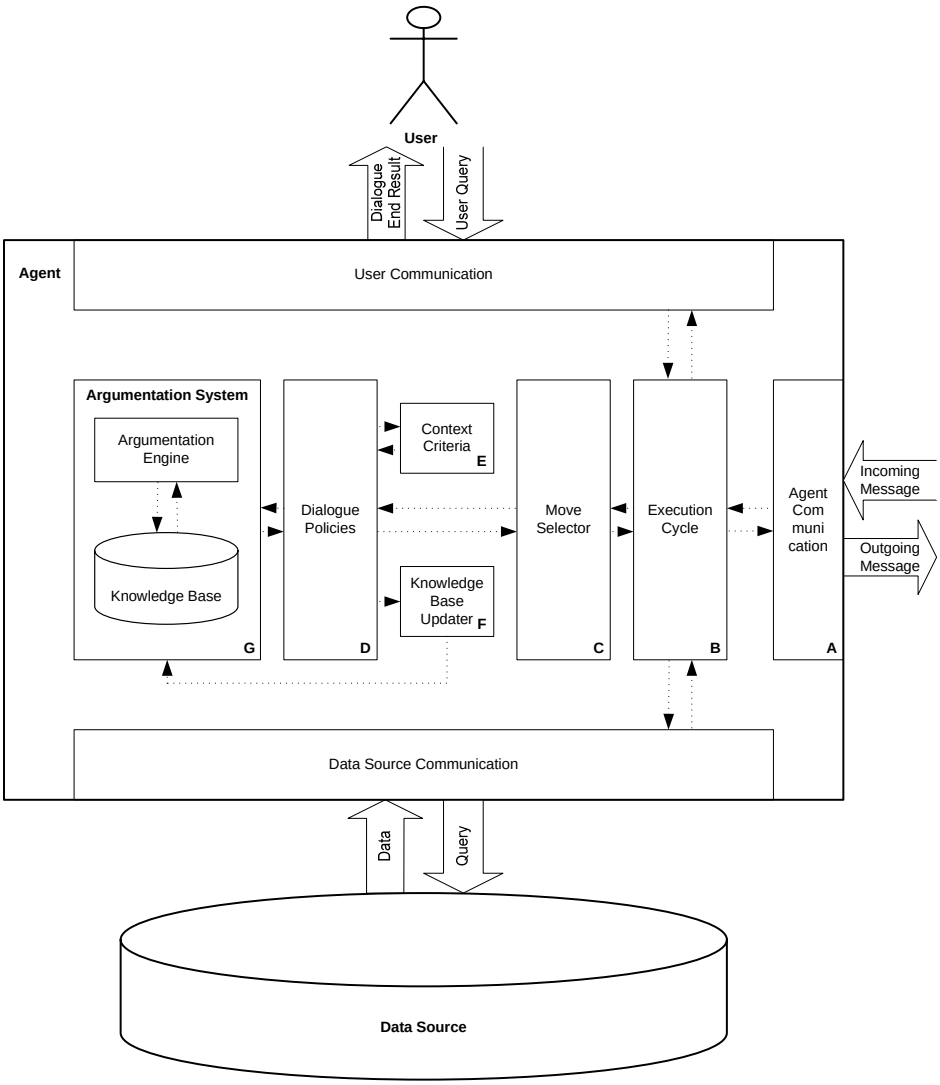

Figure 3: Individual agent architecture

D: Dialogue policies module

The dialogue policies module is called from the move selector module and prescribes the best reactions to each chosen move to fulfill the agent goals during negotiation and persuasion dialogues. When an agent is in a requesting role, he uses his dialogue policies module to try to fulfill his goal to gather intelligence data for his appointed police task. A scenario is Example 3 (page 36). In this scenario, the

responding agent initially rejects the request for intelligence data. A possible reaction could be that the requesting agent accepts the rejection. However, since the requesting agent wants to gather as much data as possible, he will react by asking for the reason for the rejection. This reaction starts an embedded persuasion dialogue where the requesting agent tries to persuade the responding agent to exchange data. When the agent is in a responding role, he uses the dialogue policies module to deliberate how such an exchange will affect his goal to protect local data and his goal to execute his appointed police task as well. For instance, in Example 1 (page 33) the responding agent could simply reject the request for data, which fulfills his goal to protect local investigations or informants. Another response is to offer the found data under the condition that the data will only be used for analysis purposes, which also promotes the goal to execute the appointed police task. In this scenario, the policy prescribes the exchange data for analysis purposes. The dialogue policies module is also used to respond to the other agent during embedded persuasion dialogues (for instance, attacking an argument).

E: Context criteria module

The dialogue policies module calls the context criteria component to decide whether arguments containing new knowledge can be accepted, challenged or preferred to other arguments. For this decision, the context criteria module applies the implemented criteria specific for the chosen application domain.

F: Knowledge base updater module

The dialogue policies module calls the knowledge updater module when knowledge must be updated. The knowledge base updater in turn adds the new knowledge to the knowledge base of the argumentation engine.

G: Argumentation system module

The argumentation system module consists of an argumentation engine and a knowledge base. The knowledge base contains the knowledge of the relevant regulations for data exchange, the local interpretations of those regulations, the goals of the agents and how actions can violate or promote those goals. The knowledge base also contains data-unit identifiers, which are references to the data units in the external database. Each agent has his own knowledge base, which is not directly accessible for other agents

6.2 Implementation

This section briefly describes the implementation of the MAS architecture for regulated data exchange between CIE officers in the Dutch police domain.

The functionality of the components from the architecture is partly implemented. The dialogue policies, knowledge base updater and context criteria modules are implemented in Java as a set of forward chaining if-then rules. The execution cycle module is implemented in Java as a while loop. For the argumentation system component, the argumentation engine is implemented by embedding the ASPIC inference engine (see subsection 4.1.1 on page 53) as a Java object. The ASPIC inference engine can be downloaded from http://aspic.cossac.org/components.html. Recall from section 5.1 that the negation policy uses abductive reasoning to find a condition under which a new offer can be made. However, the ASPIC inference engine does not support abductive reasoning; for that reason this is done manually in the current implementation. The knowledge base of the argumentation system component is implemented as an ASCII file, which specifies the rules and factual statements in the prolog-like syntax of ASPIC (see Appendix 1 on page 159).

Let us illustrate the implementation with the start of the dialogue presented in Example 8 from page 113 (which in turn is based on Example 2: the terrorist assault on the queen example). The dialogue starts with the request from agent `p` to agent `o` to give him intelligence data about a terrorist group:

```
p1:     request(p, o, exchangeData(o, p, terrorists, DATA_UNIT_ID),)
```

The responding agent processes the incoming message as follows. The execution cycle module calls the communication module, which returns the incoming message parsed as a move. Since the performative of the move is a `request`, the execution cycle module calls the data source communication component to retrieve a data-unit identifier matching the request. In this case the data source communication module returns `3` as the matching data-unit identifier, and the execution cycle instantiates the move as

```
request(p, o, exchangeData(o, p, terrorists, 3),).
```

After the instantiation, the execution cycle module calls the move selector module. The move selector selects the instantiated move as the move to be replied to and calls the dialogue policies module. The ASPIC reasoner is in turn called by the dialogue policies module to determine whether the reasoner can create a justified argument for

```
obliged(not(exchangeData(o, p, terrorists, 3),).
```

Figure 4 shows a graph of the justified argument that the ASPIC reasoner creates for the conclusion that it is forbidden to exchange the found data unit `3` about terrorists.

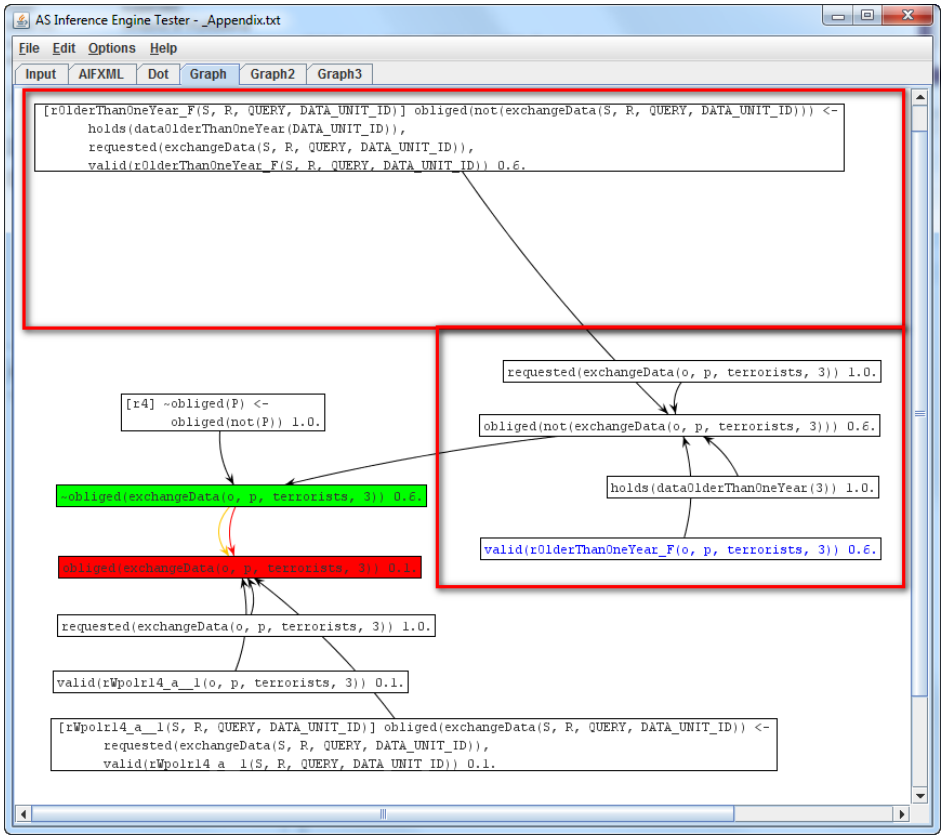

Figure 4: ASPIC reasoner

Figure 4 also illustrates that the ASPIC reasoner substitutes

`obliged(not (exchangeData(o, p, terrorists, 3)))`

with

`~obliged(exchangeData(o, p, terrorists, 3))`

using the [r4] axiom (see subsection 4.1.2 on page 61). The reasoner then determines that the argument for

`~obliged(exchangeData(o, p, terrorists, 3),)`

successfully rebuts the argument for

```
obliged(exchangeData(o, p, terrorists, 3),)
```

as illustrated by the green marking in Figure 4. As a results, the ASPIC reasoner returns ‘yes’ to the policy call (indicating that it is possible to create a justified argument). Therefore, the dialogue policies module (see step 2.3 on page 90) returns a reject move to the move selector module. The execution cycle module then calls the agent communication module to send the reject move the other agent.

```
o2:     reject(o, p, exchangeData(o, p, terrorists, 3),)
```

6.3 Chapter summary

This chapter presented the specification and implementation of the multi-agent system architecture for the Dutch police domain. Section 6.1 presented the design of the multi-agent system and individual agent architecture. The architecture abstracts terminology specific to Dutch police domain in order for the architecture to be generalizable to other domains. Section 6.2 discussed a partial implementation of the agent architecture. Subsection 8.2.1 will discuss whether the proposed multi-agent system can genuinely support automated regulated data exchange in practice. First, however, the next chapter will discuss related research on the subject of automated support of regulated data exchange.

Chapter 7

Related research

This chapter discusses related research on the topic of agents engaging in argumentation dialogues to regulate data exchange. Section 7.1 discusses two papers of Parsons, Wooldridge and Amgoud (2002; 2003), since they also investigated policies for argumentation dialogues. Section 7.2 discusses the research of Perrussel et al. (2007), because they also view interactions about exchanging data as argumentation. Similar to Perrussel et al., Doutre, McBurney and Wooldridge (2005) model regulated data exchange in the medical domain as argumentation. However, Doutre et al. (2005) is a research abstract, which does not provide sufficient details for a comparison. Therefore, Doutre et al.'s research will not be further discussed in this chapter. Section 7.3 discusses the research of Buchanan et al. (2010), because they have a different approach to support automated regulated data exchange. (i.e., instead of formalizing the written regulations and the legal reasoning process , Buchanan et al. propose to mimic its outcomes).

7.1 Parsons, Wooldridge, and Amgoud

Parsons, Wooldridge, and Amgoud (2002; 2003) analyzed information-seeking, inquiry and persuasion dialogues from the typology of dialogue types defined by Walton and Krabbe (1995). The authors specified protocols for the analyzed dialogue types and examined some properties of each protocol. Furthermore, Parsons et al. defined a set of performatives and dialogue policies which relate what arguments an agent can construct and what performatives he can make. Parsons et al. were the first who studied formal dialogue policies for argumentation dialogues and called them "agent attitudes". To compare Parson's dialogue policies with the policies of the present research, let us first investigate the differences in the underlying argument and dialogue game systems for persuasion dialogues.

The main difference between the argumentation systems is the used logical language: Parsons et al.'s system uses a propositional logic language, while the system proposed in this thesis uses ASPIC as logical language. Contrary to the

propositional language of Parsons et al., the ASPIC language supports defeasible rules (see subsection 4.1.1 on page 53). With deductive rules only premises can be attacked. Furthermore, when premises of an argument (constructed with deductive rules) are conceded then the conclusion must be conceded too. In the police domain, it is necessary that conclusions of arguments and domain rules used to create arguments can be attacked (see Example 2: the terrorist assault on the queen on page 34); these requirements are supported by the ASPIC language (see subsection 4.1.1 on page 53). ASPIC also supports variables, which enables reasoning about the validity of rules. A minor difference is the underlying semantics argumentation logic. In Parsons et al. the acceptability of individual arguments is defined with grounded semantics, while the present research also allows for preferred semantics (see the discussion about the supported argument games by ASPIC on page 61).

Let us next compare the communication language of Parsons et al.'s dialogue system with the present system. Table 5 compares the speech acts of Parsons et al.'s persuasion system with the persuasion speech acts of the proposed system and Figure 5 specifies the persuasion protocol of Parsons et al. The communication language of Parsons et al. consists of *assert*, *accept*, and *challenge* performatives. Both *assert* and *accept* performatives can be used with propositions (denoted by *P*) and with sets of propositions (denoted by *S*). Though the communication language of Parsons et al. does not explicitly define arguments, their *assert*(*S*) speech act has a meaning similar to the *argue*(*Arg*) speech act of the system of this thesis. When looking at Parsons et al's persuasion protocol described in Figure 5, it can be observed in step 4 that an agent can only assert *S* when a previous assertion of a proposition *P* is challenged. Since the combination of *assert*(*P*) and *assert*(*S*) is built in a similar way as *claim*(*P*) and *argue*(*Arg*), *assert*(*P*) can be regarded as similar to *claim*(*P*) and *assert*(*S*) regarded as similar to *argue*(*Arg*). In contrast to Parsons et al.'s system, the communication language of the present framework does not allow for accepting sets of premises with a single speech act. However, the present system's persuasion policies can evaluate all the premises of the speech act within a single turn. Therefore this speech act is not required in the present research. Furthermore, Parsons et al.'s persuasion protocol does not utilize their *accep*t(*S*) speech act, as a result this speech act is not used in their dialogues.

Table 5: Comparison of the persuasion communication languages

Parsons et al. Framework		**Present Framework**
assert(P)	≈	*claim*(*P*)
assert(S)	≈	*argue(Arg)*
accept(P)	≈	*concede(P)*
accept(S)	≈	
challenge(P)	≈	*why*(*P*)

1. *A asserts P*.
2. *B accepts P* if his acceptance attitudes allows, if not *B assert* ¬*P* if he is allowed to, or otherwise he *challenges P*.
3. If *B asserts* ¬*P*, then goto 2 with the agent roles reversed and ¬*P* in place of *P*.
4. If *B* has challenged, then:
    a. *A* asserts *S*, the support for *P*;
    b. Goto 2 for each $s \in S$ in turn.
5. *B* accepts *P* if his acceptance attitude allows, or the dialogue terminates.

Figure 5: Specification of the persuasion protocol of Parsons et al.

Let us next compare the protocol design of Parsons et al.'s (see Figure 5) and the present system. As discussed in chapter 4 on page 70, the system proposed in this thesis uses a multi-move and multi-reply protocol. Parsons et al. also uses a multi-move and multi-reply protocol, though in Parsons et al. those properties are only applicable to replies to *assert*(*S*) moves. For example, if a player asserts *S*, then the other player makes a move for every $S' \in S$ (see step 4.b of their persuasion protocol) before the turn shifts, therefore their system uses a multi-move protocol. Furthermore, since the assert *S* is a single move which is replied to with moves for every $S' \in S$, their protocol is designed as a multi-reply protocol. However, the proposed protocol allows for alternative replies to the same claim or argument, where in Parsons et al.'s protocol a player has only one chance to respond to the same claim or premise. Note that the notion of who is on the winning side is not used in the protocol of Parsons et al., thus determining whether their protocol is immediate or non-immediate reply is not applicable.

Now the dialogue policies of both systems can be compared. Parsons et al. distinguish two types of agent attitudes for persuasion: assertion attitudes determine if something can be asserted (or claimed or argued in the system of this thesis) and acceptance attitudes determine whether something can be accepted (or conceded in the present system). An agent can have one of the following three assertion attitudes.

- A *confident* agent can assert any proposition P for which he can construct an argument.
- A *careful* agent can assert any proposition P for which he can construct an argument and he cannot construct a stronger argument for $\neg P$.
- A *thoughtful* agent can assert any proposition P for which he can construct a justified argument.

An agent can have one of following the three acceptance attitudes.

- A *credulous* agent can accept proposition P for which he can construct an argument.
- A *cautious* agent can accept any proposition P for which he can construct an argument and he cannot construct a stronger argument for $\neg P$.
- A *thoughtful* agent can accept any proposition P for which he can construct a justified argument.

In comparing the dialogue policies of Parsons et al. with the policies of the present research, the following differences can be noted. First Parsons et al. only define policies for persuasion dialogues, while in the present approach also defines policies for negotiation dialogues. In addition Parsons et al.'s persuasion policies are incorporated in their protocols, while in the present research they part of the agent design. By keeping the dialogue policies part of the agent design, the proposed protocol does not refer to an agent's internal state, which enables to verify an agent's allowed moves externally (Walton and Krabbe 1995). Moreover the persuasion policies of the present research allow for more complex dialogues since they have a richer logic and communication language. Furthermore, in contrast to the dialogue policies of Parsons et al., the policies of the present research allow for domain-specific context criteria to determine whether arguments can be accepted, challenged or preferred to other arguments (see section 5.3 on page 111). An example from the Dutch police domain illustrating the need to allow for domain-specific context criteria is that arguments originating from police officers with a high authority are always accepted. What is more, as described in Prakken (2006), dialogues based on the system of Parsons et al. tend to be one-sided since only one player can build his arguments.

Finally when a policy requires an agent to construct arguments, in the present research an agent only reasons with his initial knowledge base plus the propositions that he has explicitly conceded to in the persuasion dialogue, while in Parsons et al. an agent must also reason with everything the other agent has said, regardless of whether he has conceded to this or not. Therefore, in the system of Parsons et al. it is possible to force another agent to reason with any proposition, which is impossible with the dialogue policies of the present research. For example, assume that both players have a thoughtful attitude and that the proponent p has {b <- ¬a. ¬a.} as his knowledge base and the opponent o has {a.} as his knowledge base. Now consider the following start of a dialogue between p and o.

p_1: claim b

The best move for the opponent in step o_2 is why b (or challenge b in the communication language of Parsons et al.) since this move will reveal that p uses ¬a, which conflicts with o's knowledge base. However, at this stage opponent o has to reason with his own knowledge base conjoined with the commitments of proponent p ({a.} ∪ {b.}). As a response the initial claim, opponent o applies step 2 of Parsons et al.'s persuasion protocol, and has to concede b because o has a thoughtful acceptance attitude and can create a justified argument for b. {b.}. Furthermore, it is easy to verify that if o has a thoughtful acceptance attitude then it also holds that o has to concede b if he has a cautious or credulous attitude.

$?o_2$: concede b

When the persuasion policies of the approach the present research are applied (assuming that o will not accept the claim based on his context criteria), o continues with policy step 3.4 and in this step o makes the preferred move:

O_2: why b

To summarize the comparison with Parsons et al., the present research is more extensive because of the richer logic and language, and because of the additional policies for negotiation. Moreover, the present research allows for agents to decide whether new knowledge can be accepted using domain-specific context criteria. Finally the present protocol is more flexible because it allows agents to explore alternative moves and it avoids the one-sidedness of Parsons et al.'s persuasion protocol.

7.2 Perrussel et al.

Perrussel et al. (2007) also view agent interactions about gaining access to sensitive data as argumentation dialogues. Similar to the present research, they model how two agents, a server (responding agent) and a client (requesting agent), try to

persuade each other whether data can be exchanged in cases when a server refuses a request.

Let us investigate the differences between the present research and Perrussel et al. The first difference is the way the interactions between agents are viewed. As presented in section 3.3 (page 49), interactions are modeled starting as a negotiation, which may shift to embedded persuasion if the requesting agent tries to persuade the responding agent that he is wrong about a rejection. When the negotiation terminates successfully, it is followed up with an information-seeking dialogue. Perrussel et al. view the interactions starting as information-seeking, which may shift to persuasion, and eventually shifts back to information-seeking. In contrast to Perrussel et al., the interaction model of the present research leaves room for negotiation, including the possibility of stating a condition to accept a request.

The second difference is the specification of arguments, Perrussel et al. leave the internal structure of arguments unspecified, while the present research applies ASPIC as a logic for defeasible argumentation and as knowledge representation language to define and reason with arguments. For the present research the structure of arguments needs to be specified, since the modeling of data exchange in the Dutch police domain required an analysis of permissions and the relations between goals, actions and possible consequences other goals. In the present research those permissions, goals and actions are explicitly modeled in the ASPIC language. Furthermore, the argumentation framework of Perrussel et al. requires that agents share the same set of arguments, while in the present research agents have individual knowledge bases from which they can construct their own arguments.

The third difference is the way the moves are specified. To illustrate the differences, each dialogue move presented in Perrussel et al. is compared with a similar move from the present research. As for notation, *X* indicates the requesting agent, *Y* indicates the responding agent and *DATA_UNIT_ID* denotes the data-unit identifier. Note that to denote the actual data unit corresponding with the data-unit identifier, Perrussel et al. use <content *DATA_UNIT_ID* >, while the present research uses *DATA_UNIT*. Furthermore, in the move notation *I* denotes the permission to have access to the data unit, elements *A* and *C* denote the action and the condition, and *Arg* denotes an argument. In the comparison of similar moves, the first bullet describes the meaning of the move in Perrussel et al.'s system, and the second bullet specifies the move with a similar meaning from the present research.

openDialogue(*X*, *Y*) ≈

- Requesting agent *X* indicates to responding agent *Y* that he wants to start a dialogue.

- In the present research no move is defined for opening a dialogue, since a request move implicitly initiates a dialogue.

ask(*X*, *Y*, *DATA_UNIT_ID*) ≈ *request*(*X*, *Y*, *A*, *C*)

- Requesting agent *X* asks responding agent *Y* to provide him with data.
- Requesting agent *X* asks responding agent *Y* to perform action *A* (for example provide him with data), under condition *C*.

tell(*Y*, *X*, <content *DATA_UNIT_ID* >) ≈ *sendData*(*Y*, *X*, *DATA_UNIT*)

- Responding agent *Y* provides requesting agent *X* the actual data unit corresponding with data-unit identifier *DATA_UNIT_ID*.
- Responding agent *Y* provides the data unit to requesting agent *X*.

dontTell(*Y*, *X*, *DATA_UNIT_ID*) ≈ *reject*(*Y*, *X*, *A*, *C*)

- Responding agent *Y* indicates to requesting agent *X* that he cannot provide agent *X* with data corresponding with *DATA_UNIT_ID*.
- Responding agent *Y* rejects the request or offer from requesting agent *X* to perform action *A* (for example, provide him with data) under condition *C*.

endDialogue(*Y*, *X*) ≈ *withdraw*

- Responding agent *Y* indicates to requesting agent *X* that he wants to leave the dialogue.
- An agent indicates he wants to leave the dialogue.

argue(*X*, *Y*, *I*, *Arg*, *DATA_UNIT_ID*) ≈ *argue*(*X*, *Y*, *Arg*)

- Requesting agent *X* gives to responding agent *Y* an argument *Arg* stating why the permission to access *DATA_UNIT_ID* should be equal to value *I*, where *I* can have value 1 or 0.
- Requesting agent *X* states argument *Arg* to responding agent *Y*.

Perrussel et al. have no counterparts for the following moves: *claim*(*X*, *Y*, *P*), *why*(*X*, *Y*, *P*), *concede*(*X*, *Y*, *P*), *retract*(*Y*, *X*, *P*), and *deny*(*X*, *Y*, *P*). This move comparison illustrates that the communication language of the present research is more expressive than Perrussel et al.'s research.

To illustrate how moves with arguments can be represented in both dialogue systems, consider the following example: agent *o* gives the argument to agent *p* that

it is not permitted to exchange Robert's health record, because Robert has given his Doctor limited consent to exchange his health information. With the system of Perrussel et al., this example can be represented as:

argue(*o*, *p*, *0*, Arg_1, *robert's health record*)

Where Arg_1 informally denotes: *"Robert has only given his doctor limited consent to exchange his health record information"*. Note that in Perrussel et al. the internal structure of Arg_1 cannot be specified explicitly.

With the system proposed in this research, this example can be represented as:

argue(*o*, *p*,
obliged(*not*(*exchangeData*(*o, p, robert's health record, 2*))).

obliged(*not*(*exchangeData*(*X, Y, QUERY, DATA_UNIT_ID*))
<-

holds(limitedConsent(*DATA_UNIT_ID, PATIENT, DOCTOR*)),
requested(exchangeData(*X, Y, QUERY, DATA_UNIT_ID*)),
valid(rLimitedConsent(*X, Y, QUERY,DATA_UNIT_ID,*
PATIENT,DOCTOR)).

holds(limitedConsent(*2, robert, o*)).
requested(exchangeData(*o, p, robert's health record, 2*)).
valid(*rLimitedConsent*(*o,p,robert's health record,2,robert,o*))).
)

Where '*obliged*(*not*(*exchangeData*(*o, p, robert's health record, 2*))).' denotes that it is forbidden for agent *o* to exchange the data unit *2* matching query '*robert's health record*' with agent *p.*

In the two representations of the example, several differences in the move specification can be observed. First, in Perrussel et al. the persuasion topic is fixed to convincing the other agent about the permission to access data. For example, *argue*(*o, p, 1, Arg, d*) represents that agent *o* gives argument *Arg* to agent *p* why he should have permission to access *d*, while *argue*(*p, o, 0, Arg, d*) denotes that agent *p* gives argument *Arg* to agent *o* why he cannot have permission to access *d*. In the present research, the persuasion topic is not restricted to convincing the other agent about the permission to access data (any action is possible) and other deontic modalities besides permission can be expressed. Moreover, since Perrussel et al. do not specify the internal structure of arguments, no moves related to the internal

structure are listed in their paper. The present research does specify moves related to argument structure, which allows attacking or surrendering to premises of arguments.

The fourth difference is the protocol design. The protocol of Perrussel et al. can be summarized as follows. When an agent wants to initiate a dialogue to obtain data from another agent, he starts with making an *openDialogue* move followed with an *ask* move, When a responding agent receives an *ask* move and controls the requested data, he will exchange the data using the *tell* move if the requesting agent has permission to access the data. After the responding agent has provided data, he closes the dialogue by making the *endDialogue* move. When a responding agent receives an *ask* move and does not control the requested data, he will end the dialogue using the *endDialogue* move. When a responding agent receives an *ask* move and controls the requested data, he refuses to exchange the data using the *dontTell* move if the requesting agent has no permission to access the data. After the refusal of the responding agent, he has to motivate his refusal, which starts the argumentation stage of the dialogue. During the argumentation stage, the agent to-move gives all the counter arguments for the received argument(s) for every turn. If one agent cannot counterattack anymore, the other agent has won. When the requesting agent has won, the responding agent changes the permission and exchanges the requested data.

Similar to the system of the present research, Perrussel et al.'s system uses a multi-move and multi-reply reply protocol. For instance, every dialogue according to the system of Perrussel et al. begins with the requesting agent making two moves before the turn shifts, therefore their system uses a multi-move protocol. Furthermore, their system utilizes a multi-reply protocol since agents give all counterargument moves for a received argument. Note that the notion of who is on the winning side in a dialogue is not applied in the protocol of Perrussel et al., therefore nothing can be said about whether their protocol is immediate or non-immediate reply.

The fifth difference is that, contrary to Perrussel et al., the system of this research allows agents to learn from each other. In Perrussel et al. agents share all arguments, while in system proposed in this research agents can update their knowledge bases after concede moves. To illustrate the importance of this feature, recall Example 2: the terrorist assault on the queen from page 34. In this example the responding agent initially refuses to exchange data. However, after the responding agent learns from the requesting agent that the request is an exceptional situation, he offers the requested data.

The sixth difference is that only the present research allows for domain-specific context criteria to determine whether arguments can be accepted, challenged or preferred to other arguments (see section 5.3 on page 111).

The seventh difference is that Perrussel et al. do not address dialogue policies. Their protocol even seems to be deterministic; in that case there is no point having dialogue policies since they are used to specify the best choice from the moves allowed by the protocol.

To summarize, both approaches view agent interactions to gain access to sensitive data as argumentation dialogues. However, the present research allows for negotiation and stating conditions during dialogues. As for argument specification, Perrussel et al. leave the internal structure of arguments unspecified, while the present research applies ASPIC as a logic and knowledge representation language to specify goals, actions and permissions. Furthermore, the system proposed in this research uses a richer communication language and specifies dialogue policies, which are absent in Perrussel et al. Therefore, more elaborate discussions are allowed, such as exchanging counter-counter arguments about a subconclusion. Moreover, in the system of this research, the persuasion topic is not restricted to convincing the other agent about having permission to access data. Finally, in contrast to Perrussel et al., the system proposed in this research allows agents to learn from each other and to use domain-specific context criteria determine whether arguments can be accepted, challenged or preferred to other arguments.

7.3 Buchanan et al.

Buchanan et al. (2010) also observe that regulated data exchange between organizations is often suboptimal. In theory, the regulation of data exchange should ensure that the privacy of the persons who are the subjects of the data is protected and that an optimal balance can be found between the different goals the organizations try to achieve. However, in practice this is not always realized because the norms from regulations are not well known and local policies are more aimed at the protection of local data, which results in not enough legitimate and too many illegitimate data exchanges. In Scotland, the norms expressing the necessity for organizations to exchange data are not well known because acts such as the Health and Social Care Act 2001, Police Reform Act 2002, Community Care Act 2003 and the Children Act 2004 are too unspecific for practical use for not legally trained personnel. Information-sharing agreements are used by the data exchanging organizations to operationalize the law. However, the information-sharing agreements strive to cover every possible situation and have become so complex and detailed that they are rarely used by the Scottish organizations. Furthermore,

the regulations give only vague guidance on how to balance competing goals. Buchanan et al. also give an example, which nicely illustrates how goals can conflict. In the example, a social worker, trying to establish a trust relation with a client will be hesitant to pass on information about low level drug dealing to the police, if he fears that the information will result police in activity that would make his work impossible (Richardson and Asthana 2006).

Buchanan et al. propose an information-sharing framework to support regulated data exchange. In this framework, they use the concept of single point of contact (SPoC) to serve as a data gateway between organizations, which is implemented as a SPoC agent. A SPoC agent is similar to a responding agent in the system of this research in that both agents validate requests. Moreover, both systems support regulatory compliance, since they validate requests based on domain rules derived from the legal framework and local policies and agreements. For their information-sharing framework Buchanan et al. propose that exchanging organizations have an information-sharing agreement which defines a role-based architecture and a syntax for information requests, where SPoC (Single Point-of-Contact) agents control the exchange of data. The roles identify who need access to information in order to fulfill their tasks and the SPoC agents validate requests using the domain rules defined by information-sharing agreements.

Lessig (1999) popularized the idea to let computer software force users to act according the applicable regulations by embedding legal concepts directly into the code (i.e., the source code of computer software). However, this approach only works in domains where the regulations have few open-textured terms and give few discretionary powers. Instead of embedding legal concepts directly into code or modeling the written regulations and the legal reasoning process used in the balancing act between the different goals each organization tries to fulfill, Buchanan et al. propose to mimic its outcomes. The outcomes are domain rules for specific classes of situations and are the result of the balancing act for those types of situations. The outcomes are acquired using empirical methods such surveying personnel and those outcomes are formalized using the syntax of an information-sharing agreement. Buchanan et al. state that the representations need to be validated by legal experts to be consistent with the legal framework. Their reason for choosing to model the outcomes of the legal reasoning process is that - in the short term - it can have a greater impact than explicit modeling of the legal reasoning process. Buchanan et al.'s approach is to use context levels for a request, where the information-sharing agreement defines the rights based on the context of the request. Buchanan et al. give the example that the rights to data will be higher within the context of a missing persons query than for a trivial access to data. Buchanan et al. use their survey-based approach to determine the access rights based on context.

Their approach on how to support the balancing act differs from the present research. Buchanan et al. incorporate outcomes of the balancing act in the representation of the information-sharing agreement between organizations itself. That is, before actual data exchanges, organizations are asked which contexts give access rights to which types of data. In the present research, each agent has his own rules and can negotiate and argue with other agents why data can or cannot be exchanged. Furthermore, in contrast to Buchanan et al., the balancing act is not represented in rules, but in the negotiation and persuasion policies of an agent, where the agent internally reasons about the consequences of (not) exchanging data.

To illustrate that the approach of finding a balance between different goals in the interaction can be a better approach, let us use the system of this research to model the example from the introduction of the social worker who is trying to establish a trust relation with a client. If the police officer is hesitant to give the context for his request (for example, because he does not want to unnecessarily inform other people that the requested data is of national importance), he could first try to simply request the data. If the data was not related to drug dealing, he would easily obtain the data and the social worker would not be unnecessarily informed about the officer's investigation. Let us next assume that the social worker initially refuses to exchange the sensitive data about his client, since he wants to establish a trust relation. At this point, the police officer can try to persuade the social worker that he needs the data for a case of national importance. Assuming that the social worker is persuaded by the argument, he then can also state conditions to minimize the possible negative consequences of the exchange. Let us next apply the approach of Buchanan et al. to model the example from the introduction. If the information-sharing agreement states that social workers do not exchange sensitive data with police officers, then with the approach of Buchanan et al. the requested data would not be exchanged. If the information-sharing agreement states that social workers exchange sensitive data in case of national importance, then they would always be informed about those cases. In the system of this research, the balance between goals is ultimately found in the interaction. In the example, a social worker will only be informed about national importance cases when he refuses, and sensitive data will only be exchanged in exceptional situations such as national importance.

Let us next discuss how the criminal investigations units currently balance their goals in the Dutch police domain and compare this with how the balancing act could be improved with the approach of Buchanan et al. and with the approach of this research. In the current situation, police departments use data usage codes (see subsection 2.4.3) to determine the allowed usage of data. However, the data usage codes are applied in such a way that protected data will never be automatically exchanged using the information systems, while sometimes protected data can be

exchanged if CIE officers engage in dialogues (see the examples described in section 2.5). Thus in practice, the balancing act is geared towards the goal to protect local investigations and informants

If Buchanan et al.'s approach is applied to support regulated data exchange in the Dutch police domain, then information-sharing agreements need be used to specify the types of situations in which protected data can be exchanged. Assuming that the information-sharing agreements are the outcomes of a fair and serious balancing act between different goals the criminal investigation units try to fulfill, Buchanan et al.'s approach could be used to support regulated data exchange. However, in practice it is not possible to specify every possible type of situation beforehand.

If the approach of this research is used to support regulated data exchange, the balancing act is not fixed as with the automated exchange in current situation and with the approach of Buchanan et al., but is found in the interaction. Finding the balance in the interaction can result in better support of regulated data exchange, because it allows the agents to reason with their own knowledge to show that an agreed rule based on a typical situation is not applicable in a specific situation. For example, with Buchanan et al.'s approach it is not possible to give arguments based on knowledge which are not known by the other agent. A second example is the ability to state conditions to be fulfilled by the requesting agent, which is also not possible with Buchanan et al.'s approach.

To summarize, Buchanan et al., also observe that in practice regulated data exchange between organizations is often suboptimal because the norms from regulations are not well known and local policies are more aimed at the protection of local data. Buchanan et al.'s approach to support regulated data exchange is to propose an information-sharing framework. In this framework, data exchanging organizations have an information-sharing agreement, where SPoC (Single Point-of-Contact) agents are used to control the exchange of data. A SPoC agent can be compared with a responding agent used the system of this research in that they both validate requests. Buchanan et al.'s approach on how to support the balancing act differs from the approach of the present research. Buchanan et al. incorporate the outcomes of the balancing act between different goals the organizations try to fulfill with exchanging data in the representation of the information-sharing agreement itself, while in the present research the negotiation and persuasion policies are used to assist the agents to balance their goals in their interactions. Furthermore, in contrast to Buchanan et al., each agent has his own rules and reasons about the consequences of (not) exchanging data and can negotiate and argue with other agents whether data can or cannot be exchanged. Therefore, with the approach of this research the balancing act is not fixed as with the approach of Buchanan et al., but is found in the interaction. Finally, the comparison of the approaches of the

present research with Buchanan et al.'s research illustrated how finding the balance in the interaction can result in better support of regulated data exchange.

7.4 Chapter summary

This chapter compared related research on the subject of automated support of regulated data exchange. Similar to Perrussel et al. (2007), the present research views agent interactions to gain access to sensitive data as argumentation dialogues. However, the present research proposed a new way to model the interactions as negotiation dialogues with embedded persuasion dialogues. An important contribution of this research has been the specification of realistic dialogue policies to help agents to balance their goals they try to fulfill with the exchange of data. As for the specification of dialogue policies, Parsons et al. (2002; 2003) have not specified policies for negotiation dialogues and in Perrussel et al. they are absent. Supporting the balancing act in the context of regulated data exchange is only described in Buchanan et al. (2010). However, Buchanan et al. use another approach. Instead of specifying dialogue policies to find a balance in the interaction, Buchanan et al. mimic and fixate the outcomes of the balancing act for agreed types of situations. Finally, in contrast to the discussed research in this chapter, the system proposed in this research allows agents to learn from each other and to use domain-specific criteria to determine whether new knowledge can be accepted.

Chapter 8

Conclusion

The problem statement of this thesis has been if it is possible to develop a realistic and implementable architecture for a multi-agent system that can provide a basis for automated support of regulated data exchange between organizations. The support of regulated data exchange was defined as facilitating fewer unlawful data exchanges (exchanges which are forbidden do occur in practice) and more lawful data exchanges (exchanges which are permitted do not occur in practice). Section 8.1 summarizes the results based on the problem statement of this thesis. Section 8.2 discusses whether the proposed system can provide support of regulated data exchange in practice and the contributions of this research. Finally, section 8.3 presents the wider applicability of this research and discusses the need for future research.

8.1 Answers to the research questions

Answering the problem statement is divided into several steps, which each have their own specific research questions. Therefore, every research question will be answered separately below.

Chosen domain with regulated data exchange

RQ 1 How does the problem of regulated data exchange manifests itself in the Dutch police domain?

As discussed in chapter 1, the Dutch police domain provides a worthwhile example of the problem of regulated data exchange because it illustrates that not always an optimal balance is found in the exchange of data. To answer the first research question, a description and analysis was made which was based on a study of the literature, the relevant regulations and interviews with police officers. The description and analysis showed that regional police departments need to exchange data, because departments often need data held by other departments in order to solve crime cases (see sections 2.1 and 2.2). Furthermore, each regional police

department has a criminal investigations unit for gathering intelligence data on severe crime. The analysis also showed that the criminal investigation units have to balance the goal to execute their appointed police task with the goal to protect their interests at the local level (i.e., their investigations and informants), while staying within the law. Ideally, criminal investigation units find an optimal and legitimate balance of their goals by engaging in dialogues about regulated data exchange. However, this ideal is not always realized because criminal investigation units do not know all the applicable regulations and prefer their goal to protect their own their interests by refusing to exchange data.

RQ 1.b What is the legal framework for the Dutch police domain?

In order to answer research question 1.a, a description was made of the legal framework for regulated data exchange in the Dutch police domain (see sections 2.1, 2.3 and 2.4). At the time of this research, the Police Registers Act (*Wet politieregisters,* Wpolr) was applicable for the exchange of severe-crime data between Dutch criminal investigation units, which is further detailed in the Police Registers Decree (*Besluit politieregisters,* Bpolr) and the CIE regulation (*regeling CIE*)[20]. The legislator intended a free-flow of data within parts of the Dutch police organization and has created many rules which obligate the exchange of data. To enable criminal investigation units to protect their investigations and informants, the legislator also allows grounds for refusal as a legal way to refuse the exchange of data. It was the legislator's objective that in specific cases, the relevant interests are weighed against each other. However, as indicated above, the analysis of the Dutch police organization revealed that in practice the grounds for refusal are most of the time used as default instead of using them with serious deliberation in specific cases. Since the grounds of refusals are used as default, only a part of the permitted data exchanges occurs in the Dutch police domain.

The main rule promoting the exchange of data is Wpolr article 14, paragraph a, which states that it is obliged to exchange data if the other party needs it for the execution of the police task; here the police task is used as a ground to promote the exchange of data. The main rule limiting the exchange of severe-crime data is Wpolr article 13a, paragraph 3, which states that the exchange of data can be refused if necessary for the proper execution of the police task; here the police task is used as a ground to refuse to exchange data. Wpolr, article 13a, paragraph 3 specifies which grounds for refusals can be used for exceptions for Wpolr art. 14 par a. The police officer interviews showed that those exceptions in turn can also have exceptions. Those exceptions are called exceptional situations and specify in which cases exceptions can be set aside. The analysis also showed that there are different types of local interpretations of the regulations. First, there are the data

[20] See section 2.3 for a brief discussion of the changed regulations.

usage codes which determine the allowed usage of data. In addition to the data usage codes, there are explicit grounds for refusal, such as when the data is used for an internal investigation, when the data is no more up to date, or when the data is traceable to informants.

To keep the model of regulated data exchange in the Dutch police domain manageable, assumptions were made to select the relevant excerpts of the applicable regulations and their local interpretations (see subsection 2.4.1). Furthermore, to allow for easier validation, the selected applicable rules were rewritten into an intermediate representation before they were formalized.

RQ 1.b. What are the relevant interactions in this domain?

Research question 1.b concerned the typical examples of interactions between Dutch police departments. The interactions were selected based on data gathered from interviews with police officers and were used to clarify the requirements for the multi-agent system architecture (see section 2.5). The examples are interactions where agents try to persuade or negotiate with each other in order to exchange data.

RQ 1.c. What is the relation between goals and actions and possible consequences on other goals related to data exchange?

Based on the literature study and interviews with police officers, the significant relations between goals and actions and possible consequences on other goals related to data exchange were identified (see section 2.6).

A multi-agent system as model for regulated data exchange

The second research question concerned the requirements to be met by the multi-agent system architecture in order to be a model of regulated data exchange in the Dutch police practice.

RQ 2: What are the requirements for a multi-agent system suitable to support regulated data exchange in the Dutch police organization?

The requirements are that the agents must have knowledge about the domain, are able to reason with their knowledge and are able to model the typical interactions in the Dutch police domain (see section 3.3).

RQ 2.a: Which kinds of knowledge do agents need to have?

The agents must have knowledge of the relevant regulations and the local interpretations of those regulations and have knowledge of the local intelligence data. Furthermore, the agents must have knowledge of their goals and the possible consequences of their actions (see section 3.3, page 48).

RQ 2.b: Which reasoning capabilities do the agents need?

The requirements for the agent's reasoning capabilities are that they must be able to reason with their knowledge of the relevant regulations, their goals they try to realize and the likely consequences of their actions. The examples of dialogues about regulated data exchange showed that the interaction between agents often involves argumentation. Therefore, the agents must be capable of generating and evaluating arguments for and against certain claims and they must be able to update their knowledge as a result of the dialogues (see section 3.3, page 48).

RQ 2.c: Which interaction types in this domain need to be supported by the multi-agent system?

For answering research question 2.c, the typical interactions types in the problem domain were analyzed. The examples identified in chapter 2 illustrated that agents engage in dialogues to negotiate about and to persuade each other whether intelligence data can be exchanged. The analysis showed that three dialogues types are necessary to model regulated data exchange: information-seeking, negotiation and persuasion dialogues. For the police domain, the relation between the dialogue types was interpreted as starting as a negotiation dialogue, which may shift to an embedded persuasion. If the negotiation terminates successfully, an information-seeking dialogue starts and its termination also terminates the overall interaction (see section 3.3, page 49).

Formalization of the multi-agent system

The third research question concerned the formalization of the requirements identified by answering the second question.

RQ 3: How can the requirements for the multi-agent system be formalized?

For answering research question 3, each requirement is discussed individually below.

RQ 3.a: How can the internal reasoning of an agent be formalized?

The internal reasoning was modeled as defeasible reasoning and formalized with the ASPIC argumentation system (see section 4.1). ASPIC was chosen because it supports defeasible reasoning and since the ASPIC inference engine can be used in the proof-of-concept implementation.

RQ 3.b: How can the knowledge of an agent be represented?

ASPIC was also used as a knowledge representation language. Since deontic modalities are not part of the ASPIC logic, they are expressed as predicates and with reification of the statements in their scope. For the same reasons the agent goals are expressed with reification (see subsection 4.1.2).

RQ 3.c: How can the agent interactions be represented?

As stated above, the agent interactions are interpreted as negotiation which can shift to embedded persuasion dialogues. Since this kind of embedding is modeled in Van Veenen and Prakken (2006), the dialogue system to model the interactions is based on their work. The dialogue system was adapted to be compatible with the ASPIC inference engine and to support the regulated data exchange between CIE officers (see section 4.2).

RQ 3.d: How can dialogue policies (or strategies) be defined that assist agents to balance their goals?

Two types of dialogue policies were specified for the agents, viz. for negotiation (see section 5.1) and for persuasion (see section 5.2). The negotiation policies help to balance the agent's goals when the agent is permitted to exchange data but there is a violation of his interests. An agent could simply refuse to exchange data based on the violation of interests. However, with the negotiation policy the agent tries to find a condition under which there will be no violation of his interests. If he succeeds then also his goal to execute the police task by exchanging data is fulfilled. The persuasion policies help to persuade the other agent to exchange data in case of rejection of a request for data. Since in the current police practice the grounds of refusals are used as default (see research question 1.a), helping to persuade can help to allow for more permitted data exchanges in the Dutch police domain. Furthermore, the policies model a fair and serious balancing act between assumptions like cooperativeness and protection of own interests since they are applicable in all domains where data is exchanged (see subsection 5.2.1). In the context of regulated data exchange between criminal investigation units, cooperativeness promotes the goal to execute the police task by enabling more data exchanges and the goal to protect their own interests promotes the goal to protect local investigations and informants.

RQ 3.e: Dialogues can result in knowledge updates. How can these changes in knowledge be modeled?

A reasonable policy for such updates was defined but the research revealed that with this and similar policies dialogue moves can be undermined by later knowledge updates during dialogues (see subsection 5.2.3). This problem was in this research solved with a pragmatic solution, leaving a more principled solution to future research.

Specification and implementation of the architecture

RQ 4: What is the architecture of the multi-agent system for regulated data exchange between different organizations?

The architecture of the multi-agent system was developed using typical examples of dialogues between Dutch criminal investigation units about regulated data exchange (see section 6.1).

RQ 4.a: How can the architecture of the multi-agent system be generalized to be applicable to similar domains?

In order for the architecture to be generalizable to other domains, all terminology specific to the Dutch police domain was abstracted in the architecture (see subsection 6.1.1).

RQ 4.b: How can the architecture be implemented into a proof-of-concept application for a problem domain?

The functionality was partly implemented in the Java programming language. The argumentation system engine was implemented by embedding the ASPIC inference engine in the agents (see section 6.2).

8.2 Discussion and contribution

This section discusses whether the proposed system can provide the basis for automated support of regulated data exchange in practice and it discusses the contribution of this research to the fields of multi-agent systems research, AI & Law research and argumentation theory research.

8.2.1 Applicability in practice

As stated in the first chapter, the research aim of this thesis has been to investigate how theories from multi-agent systems research, AI & Law research, and argumentation theory research can be further developed and applied in a realistic problem domain to provide the basis for automated support of organizations in promoting their goals in the context of regulated data exchange. The combination of a multi-agent system approach and argumentation approach allowed the modelling of data exchanging organizations that try to promote their goals by engaging in dialogues about regulated data exchange. Multi-agent system theory has been used to model data exchanging organizations, where the agents represent the members of the organizations. Furthermore, a multi-agent system allows organizations to keep their own data. Keeping data locally is important for a system to be accepted in the Dutch police domain because the criminal investigation units want to be able to control their data (see section 3.2). Argumentation theory has been used to model dialogues and the internal reasoning of the software agents. The approach of this research to allow organizations to promote their goals was to develop policies for the agent dialogues about regulated data exchange. The dialogue policies model a fair and serious balancing act between assumptions like cooperativeness and

protection of own interests. A fair and serious balancing act allows for more permitted data exchanges and fewer forbidden data exchanges, while staying within the law (see also the answer to research question 3.d).

Let us next discuss if it is possible to develop a realistic and implementable architecture for a multi-agent system that can provide the basis for automated support of regulated data exchange between organizations. By using the research described above, the multi-agent system architecture has been specified and implemented to illustrate that it can in principle support data exchange in the Dutch police domain. However, to determine whether the proposed multi-agent system can genuinely support automated regulated data exchange in the Dutch police practice, with the intended positive effects on data exchange, two issues need to be addressed.

The first issue is that it is not known how a system based on the model of this research will be used in practice. The second issue is that the implementation of the multi-agent system is dependent on the quality of the formalization of the applicable domain knowledge.

With regards to the first issue, the system's policies model a fair balancing act, while in practice CIE officers are geared towards the protection of data. Therefore, while the system allows for more permitted data exchanges based on a fair balancing act, the police officers could interpret this as undesirable. This interpretation can result in police officers not accepting the system or using the multi-agent system in an unintended way. For example, CIE officers could try to circumvent the system by using informal ways to exchange data. Moreover, the multi-agent system also shows which data exchanges are forbidden, which could be interpreted in practice as being too restrictive, which also can result in circumventing or not accepting the system. A solution to this problem probably requires organizational measures and additional training of police officers. A study of such solutions is clearly outside the scope of the present research.

Besides the actual usage in practice, the system's ability to provide the basis of automated support of regulated data exchange is also dependent on the formalized domain knowledge elicited from the legal sources and their local interpretations. Moreover, this formalized knowledge needs to be maintained. Making domain knowledge explicit and formalize it in a language executable by a software system is a difficult task, and is known as the knowledge-acquisition bottleneck (Feigenbaum 1984). The knowledge-acquisition bottleneck is an issue in several stages in creating a model of regulated data exchange in the Dutch police domain. The first stage is the acquisition and formalization of the applicable regulations. For this research assumptions (see subsection 2.4.1 on page 18) were made to keep the

model of regulated data exchange in the Dutch police domain manageable and generalizable. For an application in practice, the scope of the model needs to be extended, which involves formalizing more applicable regulations and validation by domain experts. The second stage is the acquisition and formalization of the local interpretations. In the Dutch police domain, many local rules are available informally only (i.e., undocumented knowledge that sits in the heads of CIE officers and is not written into local rules). When acquiring informal knowledge, the CIE-officers may provide incomplete or incorrect knowledge and they may not be able to explicate their knowledge. Therefore, user experiments in realistic scenarios are necessary to address the above mentioned issues and to determine how an implementation of the system in practice can support regulated exchange in practice.

8.2.2 Contributions to research

This thesis contributes to the fields of multi-agent systems, argumentation, and AI & Law in their combination and application in a serious problem domain. The multi-agent system architecture combines and adapts several elements from the literature: defeasible-argumentation for the agents' internal reasoning and a dialogue system to model the agent dialogues. The research has thus by way of a realistic case study provided support for the naturalness and applicability of ideas from the literature.

Furthermore, this research has proposed a novel view on the nature of dialogues in the context of regulated data exchange, i.e., as negotiation with embedded persuasion. An important contribution to the research literature has been the specification of dialogue policies to assist the agents to balance their goals. The dialogue policies were made applicable to other domains were data is exchanged by incorporating assumptions, such as cooperativeness and protection of own interests. Moreover, the policies were parameterized, which allows them to be fine tuned for the chosen application domain. To the best of my knowledge this is the first research which specified realistic negotiation and persuasion policies in the context of regulated data exchange.

Another research contribution has been the definition of realistic knowledge update policies for agents during dialogues about regulated data exchange. Research on this issue has also identified a new research issue in designing dialogue systems for persuasion, by identifying the problem that during a dialogue between agents, previous dialogue moves sometimes can no longer be maintained as a result of knowledge updates at later stages in a dialogue. In this thesis a pragmatic solution to this problem was provided, while the topic of knowledge updates during dialogues was identified as an important issue for future research.

8.3 Wider applicability and the need for future research

This section discusses the applicability of this research to other domains and describes the need for further research.

In this research, terminology specific to the Dutch police was abstracted and the dialogue policies were parameterized. Therefore, this research can be used in other domains where data exchanging organizations must balance the goal to execute their tasks with the goal to protect their interests at the local level. An example of a domain for which this research is applicable, is the exchange of patient data. Similar to criminal investigation units, hospitals may want keep their sensitive data locally and different hospitals may use a different policy towards data exchange. A reason for hospitals to keep their data local is that they are responsible of their patient data (Aldea et al. 2001). Furthermore, hospitals have to interact to achieve their goals such as providing the appropriate care to a patient (Huang et al. 1995), while complying with the applicable laws and regulations. For instance[21], assume that a traveler gets a heart attack and is taken into a hospital in the foreign country he currently visits. In order for the staff of the hospital to prescribe the appropriate drugs, they have to obtain the traveler's medical history from his doctor. However, the traveler's doctor initially refuses to exchange the requested data because the traveler has given limited consent to exchange the data. To find a balance between their goals, the hospital and the doctor can try to negotiate and persuade each other whether and on which terms the data can be exchanged. Note that the continual growth of the internet can cause an increase in the number of regulated data exchanges, which will make this research more widely applicable.

As discussed in the previous section, more research with user experiments in realistic scenarios is necessary to determine if an implementation of the multi-agent system architecture can support regulated exchange in practice. Furthermore, to improve the ability of the system to support regulated data exchange, the possibility of more extensive user interaction needs to be investigated. For example when a software agent in a requesting role cannot persuade the other software agent to exchange data, the requesting agent could query his human user to state other reasons which may help with his persuasion. A second example is when a software agent receives an unknown condition or argument and queries the user if this new knowledge can be added to the software agent's knowledge base. Note that in the present research context criteria are used to evaluate new arguments (see section 5.3). A third example is that when a human user decides to deviate from the decision made by the software agent, the software agent could ask for the reason for his decision. Possible research questions related to more extensive user interaction

[21] This example is based on the example described in Perrussel et al. (2007).

are: how can knowledge from users be acquired, and what are typical examples of more extensive user interaction? A suggestion for further research is to investigate other combination patterns of dialogue types to model regulated data exchange and compare them with the approach of this research. As stated in the previous section, this research revealed the complication that dialogue moves can be undermined by later knowledge updates during dialogues (see also subsection 5.2.3). For the present research a pragmatic solution was chosen. A suggestion for future research is to investigate this problem to provide a more systematic solution.

Concluding, this research has shown that it is possible to develop a realistic and implementable multi-agent system architecture that can provide the basis for automated regulated data exchange. For this research, theories from multi-agent systems research, AI & Law research and argumentation theory research have been further developed and applied in realistic domain. If the present research results are further developed in research based on an implementation in practice combined with the possibility of more extensive user interaction, then automated regulated data exchange between organizations in practice may well become possible and beneficial for society.

Appendix

Representation of domain rules in ASPIC

This appendix concerns the representation of the domain rules in the ASPIC logic (for a description of ASPIC, see subsection 4.1.1 on page 53). In section 2.4, the selected applicable rules were rewritten into an intermediate representation, which in turn are formalized in this appendix. In the formalization, the variable *S* denotes the responding agent, the variable *R* denotes the requesting agent, the variable *QUERY* represents the search query, the variable *CIE_OFFICER* denotes the CIE officer, the variable *INFORMANT* represents the informant, and the variable *DATA_UNIT_ID* denotes the data unit identifier, which is associated with a data unit (see subsection 6.1.1). Furthermore, every domain rule is accompanied with a factual statement matching the valid condition. To enable arguing about grounds for a rule, the factual statements about the validity can be replaced by domain rules (see also page 60).

Domain rule rWpolr14_a__1

(Wpolr14.a implication 1)

IF a CIE agent needs data for the execution of the police task,
THEN
it is obliged to exchange the data from the severe-crime register with the CIE agent.

Based on the assumptions (see page 18) that CIE officers only request data for the proper execution of the police task and that all CIE agents are authorized to receive intelligence data, Wpolr article 14, paragraph a, implication 1 (page 20) can be represented in ASPIC as:

```
[rWpolr14_a__1(S, R, QUERY, DATA_UNIT_ID)]
obliged(exchangeData(S, R, QUERY, DATA_UNIT_ID))
<-
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rWpolr14_a__1(S, R, QUERY, DATA_UNIT_ID)) 0.1.
```

```
valid(rWpolr14_a__1(S, R, QUERY, DATA_UNIT_ID)) 0.1.
```

Domain rule rWpolr13a_3__1

(Wpolr13a.3 part 1)

IF a refusal to exchange data is necessary for the proper execution of the police task,
THEN
it is permitted to refuse to exchange data.

Part 1 of Wpolr article 13a, paragraph 3 is instantiated with article 11 of the Bpolr and with the *rOlderThanOneYear_F* domain rule. Therefore, this rule is not represented in ASPIC.

Domain rule rWpolr13a_3__2

(Wpolr13a.3 part 2)

IF a refusal to exchange data is necessary for the proper execution of the police task,
THEN
it is permitted to exchange data with restricting conditions regarding to further use.

Part 2 of Wpolr article 13a, paragraph 3 is instantiated with article 11 of the Bpolr. Hence, this rule is not represented in ASPIC.

Domain rule rBpolr11_1_a

(Bpolr 11.1. a)

IF the data about informants concerns crimes which were or are going to be committed
THEN
an administrator is permitted to refuse the exchange of data.

Article 11, paragraph 1, subparagraph a, of the Bpolr is an instantiation of article 13a, paragraph 3, sentence 1, of the Wpolr for administrators of criminal investigation units. Bpolr, article 11, paragraph 1, subparagraph a (*rBpolr11_1_a*), concerns protecting data originating from informants. Data originating from informants have ‘source protection’, which is a special case of a ground for refusal. *rBpolr11_1_a* is not represented in ASPIC, but is implemented with the following

local domain rules: *rDataUsageCode300_SP*, *rSourceProtection_NP*, *rSourceProtection_F*, and *rTraceable_SP*.

Domain rule rBpolr11_1_b

(Bpolr 11.1. b)

IF *the data originates from a register of which, given the special nature of the register, in case of exchanging data, creates immediate danger for the registered person or third parties*

THEN

an administrator is permitted to refuse the exchange of data.

Article 11, paragraph 1, subparagraph b, of the Bpolr is an instantiation of article 13a, paragraph 3, sentence 1, of the Wpolr for administrators of criminal investigation units. Bpolr, article 11, paragraph a, subparagraph b (*rBpolr11_1_b*), concerns the risk of revealing the registered people or third parties. Data with risks have a 'risk factor, which is a special case of a ground for refusal. Bpolr, article 11, paragraph a, subparagraph b, is not directly represented in ASPIC, but is implemented with the following domain rules: *rDataUsageCode200_RF*, *rRiskFactor_F* and *rInternalInvestigation_RF*.

Domain rule rBpolr11_3__1

(Bpolr 11.3 sentence 1)

IF *a refusal to exchange data is not necessary for the proper execution of the police task,*

THEN

(Bpolr 11.1) is not applicable.

An assumption (see page 18) is that CIE officers only refuse to exchange data for the proper execution of the police task. Therefore, Bpolr article 11, paragraph 3, sentence 1 is not represented in ASPIC.

Domain rule rBpolr11_3__2

(Bpolr 11.3 sentence 2)

IF *data is exchanged*

THEN

it is permitted to enforce restrictions on the usage of data.

Article 11, paragraph 3, sentence 2 of the Bpolr is an instantiation of article 13a, paragraph 3, part 2 of the Wpolr for administrators of criminal investigation units. Bpolr article 11, paragraph 3, sentence 2 is interpreted in the local rules as 'can be used for analysis purposes only', and therefore serves as a justification for those local rules. The following domain rules are interpretations of Bpolr article 11, paragraph 3, sentence 2: *rDataUsageCode200A_P*, *rDataUsageCode300A_P*, *rSourceProtectionAnalysis_P*, *rRiskFactorAnalysis_P* and *rAnalysiQuery_AP*.

Domain rule rDataUsageCode11_O

Data usage code 11

IF *data has usage code 11*
THEN
it is obliged to exchange the data.

This rule is an interpretation of Wpolr article 14, paragraph a, implication 1. The domain rule states that it is obliged to exchange data with usage code 11, and be represented in ASPIC as:

```
[rDataUsageCode11_O(S, R, QUERY, DATA_UNIT_ID)]
obliged(exchangeData(S, R, QUERY, DATA_UNIT_ID))
<-
holds(dataUsageCode11(DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rDataUsageCode11_O(S, R, QUERY, DATA_UNIT_ID)) 0.5.


valid(rDataUsageCode11_O(S, R, QUERY, DATA_UNIT_ID)) 0.5.
```

Note that '`requested(exchangeData(S, R, QUERY, DATA_UNIT_ID))`' is added as a premise since determining whether an action is obliged, presupposes that the action is requested.

Domain rule rDataUsageCode200_F

Data usage code 200 forbidden

IF *data has usage code 200*
THEN
it is forbidden to use the data for operational usage

This domain rule concerns the usage of data with usage code *200* outside the criminal investigation units (cf. 'operational usage'). Based on the assumptions (see

page 18) that CIE officers do not exchange intelligence data outside criminal investigation units, this domain rule is not represented in ASPIC.

Domain rule rDataUsageCode200A_P

Data usage code 200 coordination and analysis

IF *data has usage code 200*
THEN
it is permitted use the data only under certain conditions for coordination and analysis purposes

This domain rule is interpretation of Bpolr article 11, paragraph 3, sentence 2 (*rBolr11_3__2*) and allows data with data usage code *200* to be used for analysis purposes. The domain rule can be represented in ASPIC as:

```
[rDataUsagecode200A_P(S, R, QUERY, DATA_UNIT_ID, CIE_OFFICER)]
~obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(dataUsageCode200(DATA_UNIT_ID)),
holds(analysisPurposesOnly(CIE_OFFICER, DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rDataUsagecode200A_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER)).


valid(rDataUsagecode200A_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER)).
```

Note that '`permitted(exchangeData(S, R, QUERY, DATA_UNIT_ID))`' is substituted with '`~obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))`', see subsection 4.1.2 on page 62).

Domain rule rDataUsageCode200_RF

Data usage code 200 risk factor

IF *data has usage code 200*
THEN
the data is a risk factor

This domain rule is an interpretation of Bpolr, article 11, paragraph a, subparagraph b (*rBpolr11_1_b*) and states that data with usage code *200* is a risk factor.

```
[rDataUsagecode200_RF(DATA_UNIT_ID)]
holds(riskFactor(DATA_UNIT_ID))
<-
```

```
holds(dataUsageCode200(DATA_UNIT_ID)),
valid(rDataUsagecode200_RF(DATA_UNIT_ID)) 0.3.


valid(rDataUsagecode200_RF(DATA_UNIT_ID)) 0.3.
```

Domain rule rDataUsageCode300_F

Data usage code 300 forbidden

IF *data has usage code 300*
THEN
it is forbidden to use the data for operational usage

This domain rule concerns the usage of data with usage code *300* outside the criminal investigation units (cf. 'operational usage'). Based on the assumptions (see page 18) that CIE officers do not exchange intelligence data outside criminal investigation units, this domain rule is not represented in ASPIC.

Domain rule rDataUsageCode300A_P

Data usage code 300 coordination and analysis

IF *data has usage code 300*
THEN
it is permitted use the data only under certain conditions for coordination and analysis purposes

This domain rule is an interpretation of Bpolr article 11, paragraph 3, sentence 2 (*rBolr11_3__2*) and allows data with usage code *300* to be used for analysis purposes. The domain rule can be represented as:

```
[rDataUsagecode300A_P(S, R, QUERY, DATA_UNIT_ID, CIE_OFFICER)]
~obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(dataUsageCode300(DATA_UNIT_ID)),
holds(analysisPurposesOnly(CIE_OFFICER, DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rDataUsagecode300A_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER)).


valid(rDataUsagecode300A_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER)).
```

Note that In the formalization '`permitted(P)`' is substituted with '`~obliged(not(P))`', see subsection 4.1.2 on page 62.

Domain rule rDataUsageCode300_SP

Data usage code 300 source protection

IF *data has usage code 300*
THEN
the data has source protection

This domain rule concerns the protection of informants (cf 'source protection' described in subsection 2.4.3 on page 24). The domain is an interpretation of Bpolr, article 11, paragraph 1, subparagraph a (*rBpolr11_1_a*) and states that data with usage code *300* has source protection. This domain rule can be represented in ASPIC as:

```
[rDataUsagecode300_SP(DATA_UNIT_ID, INFORMANT)]
holds(sourceProtection(DATA_UNIT_ID, INFORMANT))
<-
holds(dataUsageCode300(DATA_UNIT_ID)),
holds(supplier(DATA_UNIT_ID, INFORMANT)),
valid(rDataUsageCode300_SP(DATA_UNIT_ID, INFORMANT)).


valid(rDataUsageCode300_SP(DATA_UNIT_ID, INFORMANT)).
```

Domain rule rSourceProtection_NP

This domain rule is an interpretation of Bpolr, article 11, paragraph 1, subparagraph a (*rBpolr11_1_a*) and states that if data has source protection and the data is exchanged, then the informant who has supplied the data has no protection (see page 37). The domain rule can be represented as:

```
[rSourceProtection_NP(S, R, QUERY, DATA_UNIT_ID, INFORMANT)]
holds(not(protection(DATA_UNIT_ID, INFORMANT)))
<-
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
happens(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rSourceProtection_NP(S, R, QUERY, DATA_UNIT_ID, INFORMANT)).


valid(rSourceProtection_NP(S, R, QUERY, DATA_UNIT_ID, INFORMANT)).
```

Domain rule rSourceProtection_F

Source protection forbidden to exchange data

IF *data has a source protection*
THEN
it is forbidden to exchange the data.

This domain rule is an interpretation of Bpolr, article 11, paragraph 1, subparagraph a (*rBpolr11_1_a*) and states that it is forbidden to exchange data with source protection. The domain rule can be represented in ASPIC as:

```
[rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT)]
obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT))
0.6.


valid(rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT))
0.6.
```

Domain rule rSourceProtectionAnalysis_P

Source protection and analysis permitted to exchange data

IF *data has a source protection and a CIE officer uses the data for analysis purposes only*

THEN

it is permitted to exchange the data.

This domain rule is an interpretation of Bpolr article 11, paragraph 3, sentence 2 (*rBolr11_3__2*) and states that data with source protection can be used for analysis purposes. The domain rule can be represented in ASPIC as:

```
[rSourceProtectionAnalysis_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER, INFORMANT)]
~obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(sourceProtection(DATA_UNIT_ID, INFORMANT)),
holds(analysisPurposesOnly(CIE_OFFICER, DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rSourceProtectionAnalysis_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER, INFORMANT)) 0.7.


valid(rSourceProtectionAnalysis_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER, INFORMANT)) 0.7.
```

Domain rule rRiskFactor_F

Risk factor forbidden to exchange

IF *data has a risk factor*

THEN

it is forbidden to exchange the data.

This domain rule is an interpretation of Bpolr, article 11, paragraph a, subparagraph b (*rBpolr11_1_b*) and states that it is forbidden to exchange data which is a risk factor. The domain rule can be represented in ASPIC as:

```
[rRiskFactor_F(S, R, QUERY, DATA_UNIT_ID)]
obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(riskFactor(DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rRiskFactor_F(S, R, QUERY, DATA_UNIT_ID)) 0.6.


valid(rRiskFactor_F(S, R, QUERY, DATA_UNIT_ID)) 0.6.
```

Domain rule rRiskFactorAnalysis_P

Risk factor and analysis permitted to exchange data

IF *data has a risk factor and are used for analysis purposes*
THEN
it is permitted to exchange the data.

This domain rule is interpretation of Bpolr article 11, paragraph 3, sentence 2 (*rBolr11_3__2*) and allows data with a risk factor to be used for analysis purposes. The domain rule can be represented in ASPIC as:

```
[rRiskFactorAnalysis_P(S, R, QUERY, DATA_UNIT_ID, CIE_OFFICER)]
~obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(riskFactor(DATA_UNIT_ID)),
holds(analysisPurposesOnly(CIE_OFFICER, DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rRiskFactorAnalysis_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER)).


valid(rRiskFactorAnalysis_P(S, R, QUERY, DATA_UNIT_ID,
CIE_OFFICER)).
```

Domain rule rTraceable_SP

Traceable source protection

IF *data is traceable to an informant*
THEN
the data originating from the informant have source protection.

This domain rule is interpretation of Bpolr, article 11, paragraph 1, subparagraph a (*rBpolr11_1_a*) and states that if data is traceable to an informant then the data originating from the informant have source protection (see page 30). This rule can be represented in ASPIC as:

```
[rTraceable_SP(DATA_UNIT_ID, INFORMANT)]
holds(sourceProtection(DATA_UNIT_ID, INFORMANT))
<-
holds(traceable(DATA_UNIT_ID, INFORMANT)),
valid(rTraceable_SP(DATA_UNIT_ID, INFORMANT)) 0.6.


valid(rTraceable_SP(DATA_UNIT_ID, INFORMANT)) 0.6.
```

Domain rule rInternalInvestigation_RF

Colleague then risk factor

IF *data is used for internal investigation about a colleague*
THEN
the data has a risk factor

This domain rule is an interpretation of Bpolr, article 11, paragraph a, subparagraph b (*rBpolr11_1_b*) and states that if data is used for an internal investigation about a colleague, then the data is a risk factor. The domain rule can be represented is ASPIC as:

```
[rInternalInvestigation_RF(DATA_UNIT_ID, CIE_OFFICER)]
holds(riskFactor(DATA_UNIT_ID))
<-
holds(internalInvestigation(DATA_UNIT_ID, CIE_OFFICER)),
valid(rInternalInvestigation_RF(DATA_UNIT_ID, CIE_OFFICER)).


valid(rInternalInvestigation_RF(DATA_UNIT_ID, CIE_OFFICER)).
```

Domain rule rOlderThanOneYear _F

Older than one year forbidden to exchange data

IF *data is older than one year*
THEN
It is forbidden to exchange the data.

This domain rule is interpretation of Wpolr article 13a, paragraph 3, part 1 (*rWpolr13a_3__1*). In the Dutch police practice, data older than one year is

considered as not up to date, and could therefore hinder the proper execution of the police task. The domain rule stating that it is forbidden to exchange requested data older than one year can be formalized as:

```
[rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)]
obliged(not(exchangeData(S, R, QUERY, DATA_UNIT_ID)))
<-
holds(dataOlderThanOneYear(DATA_UNIT_ID)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)) 0.6.


valid(rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)) 0.6.
```

Domain rules ExceptionalSituations

Exceptional situations

IF *National importance*
OR
Attempted assault on the queen
THEN
an exceptional situation occurs

In exceptional cases the rules forbidding the exchange of data can be set aside. Examples of exceptional cases are national importance and an attempted assault on the queen. Of course, other examples of exceptional situations exist. Since this domain rule covers multiple exceptional situations, every example of an exceptional situation is represented a with specific ASPIC domain rule.

An attempted assault on the queen as an example of an exceptional situation can be represented as:

```
[rQueen_ES(DATA_UNIT_ID)]
holds(exceptionalSituation(DATA_UNIT_ID))
<-
holds(attemptAssaultQueen(DATA_UNIT_ID)),
valid(rQueen_ES(DATA_UNIT_ID)) 0.8.


valid(rQueen_ES(DATA_UNIT_ID)) 0.8.
```

National importance as an example of an exceptional situation can be represented in ASPIC as:

```
[rNationalImportance_ES(DATA_UNIT_ID)]
holds(exceptionalSituation(DATA_UNIT_ID))
<-
```

```
holds(nationalImportance(DATA_UNIT_ID)),
valid(rNationalImportance_ES(DATA_UNIT_ID)) 0.8.


valid(rNationalImportance_ES(DATA_UNIT_ID)) 0.8.
```

Domain rule rSetAsideProtection

Exceptional situation set aside protection

IF *an exceptional situation occurs*

THEN

the rules forbidding the exchange of data is not applicable

Since setting aside the rules forbidding the exchange of data covers multiple domain rules, every concerned rule is represented as a separate ASPIC rule.

The domain rule stating that in an exceptional situation, the rule stating that it forbidden to exchange requested data older than one year is not applicable can be formalized in ASPIC as:

```
[rExceptionalSituation_OTOY(S, R, QUERY, DATA_UNIT_ID)]
~rOlderThanOneYear_F(S, R, QUERY, DATA_UNIT_ID)
<-
holds(exceptionalSituation(DATA_UNIT_ID)),
valid(rExceptionalSituation_OTOY(S, R, QUERY, DATA_UNIT_ID)) 0.7.


valid(rExceptionalSituation_OTOY(S, R, QUERY, DATA_UNIT_ID)) 0.7.
```

The domain rule stating that in an exceptional situation, the rule stating that it forbidden to exchange requested data which is a risk factor is not applicable can be formalized in ASPIC as:

```
[rExceptionSituation_RF(S, R, QUERY, DATA_UNIT_ID)]
~rRiskFactor_F(S, R, QUERY, DATA_UNIT_ID)
<-
holds(exceptionalSituation(DATA_UNIT_ID)),
valid(rExceptionSituation_RF(S, R, QUERY, DATA_UNIT_ID)) 0.7.


valid(rExceptionSituation_RF(S, R, QUERY, DATA_UNIT_ID)) 0.7.
```

The domain rule stating that in an exceptional situation, the rule stating that it forbidden to exchange requested data which has source protection is not applicable can be represented in ASPIC as:

```
[rExceptionSituation_SP(S, R, QUERY, DATA_UNIT_ID, INFORMANT)]
~ rSourceProtection_F(S, R, QUERY, DATA_UNIT_ID, INFORMANT)
<-
holds(exceptionalSituation(DATA_UNIT_ID)),
valid(rExceptionSituation_SP(S, R, QUERY, DATA_UNIT_ID,
INFORMANT)) 0.7.


valid(rExceptionSituation_SP(S, R, QUERY, DATA_UNIT_ID,
INFORMANT)) 0.7.
```

Domain rule rProtectionGoal

An agent goal is the protection of his local investigations and informants (see page 37). The domain rule *rProtectionGoal* states that if data of an agent originates from an informant, then the goal of the agent is to protect the informant, which can be represented in ASPIC as:

```
[rProtectionGoal(S, DATA_UNIT_ID, INFORMANT)]
goal(S, protection(DATA_UNIT_ID, INFORMANT))
<-
holds(informantOf(INFORMANT, DATA_UNIT_ID, S)),
valid(rProtectionGoal(S, DATA_UNIT_ID, INFORMANT)).


valid(rProtectionGoal(S, DATA_UNIT_ID, INFORMANT)).
```

Domain rule rAnalysisQuery_AP

> *IF a query is made for analysis purposes and data is exchanged based on that query*
>
> *THEN*
>
> *it is obliged to use the data for analysis purposes only.*

This domain rule is interpretation of Bpolr article 11, paragraph 3, sentence 2 (*rBolr11_3__2*) and states that data obtained for analysis purposes can only be used for analysis purposes, which can be represented in ASPIC as:

```
[rAnalysisQuery_AP(S, R, QUERY, DATA_UNIT_ID)]
obliged(analysisPurposesOnly(R, DATA_UNIT_ID))
<-
holds(analysisQuery(R, QUERY)),
requested(exchangeData(S, R, QUERY, DATA_UNIT_ID)),
valid(rAnalysisQuery_AP(S, R, QUERY, DATA_UNIT_ID)).


valid(rAnalysisQuery_AP(S, R, QUERY, DATA_UNIT_ID)).
```

Domain rules violationOfOwnInterestesA, rViolationOfOwnInterestsB

The violation of interests rule is related to the *rProtectionGoal* domain rule, and expresses how an agent goal is violated. The domain rule states that if something is (not) a goal of an agent but the opposite holds, then his interests are violated (see page 63), which can be represented as:

```
[rViolationOfOwnInterestsA(S, P)]
violationOfOwnInterests(S)
<-
holds(not(P)), goal(S, P).

[rViolationOfOwnInterestsB(S, P)]
violationOfOwnInterests(S)
<-
holds(P), goal(S, not(P)).
```

References

Aldea, A., Lopez, B., Moreno, A., Riano, D., and Valls, A. (2001) A multi-agent system for organ transplant coordination. *Proceedings of the 8th Conference on AI in Medicine in Europe: Artificial Intelligence Medicine*, 413 - 416, Springer-Verlag, UK

Amgoud, L., Bodenstaff, L., Caminada, M., McBurney, P., Parsons, S., Prakken, H., Van Veenen, J., and Vreeswijk, G. (2006) *Final review and report on formal argumentation system*, Deliverable D2.6, ASPIC IST-FP6-002307

Amgoud, L., Caminada, M., Cayrol, C., Doutre, S., Lagasquie, MC., Prakken, H., and Vreeswijk, G. (2004) *Theoretical framework for argumentation*, Deliverable D 2.1, ASPIC IST-FP6-002307

Amgoud, L., Maudet, N., and Parsons, S. (2000) Modelling dialogues using argumentation. *Proceedings of the Fourth International Conference on MultiAgent Systems*:31 - 38, Boston

Bench-Capon, T. J. M. (2003) Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation* 13:429 - 448.

Bench-Capon, T. J. M. and Coenen, F. P. (1992) Isomorphism and legal knowledge based systems. *Artificial Intelligence and Law* 1:1, 65 - 86.

Bench-Capon, T. J. M. and Dunne, P. E. (2007) Argumentation in artificial intelligence. *Artificial Intelligence* 171:10-15, 619 - 641.

Bench-Capon, T. J. M. and Sartor, G. (2003) A model of legal reasoning with cases incorporating theories and values. *Artificial Intelligence* 150:1-2, 97 - 143.

Bench-Capon, T. J. M. and Sergot, M. (1989) Towards a rule-based representation of open texture in law. In Walter, C. (eds.), 39 - 60, Quorum Books, New York

Bentahar, J., Maamar, Z., Benslimane, D., and Thiran, P. (2007) An argumentation framework for communities of web services. *IEEE Intelligent Systems* 22:6, 75 - 83.

Bonthuis, M. J. (2003) *Internal report: Rechtsnormen en praktijk bij de Criminele Inlichtingen Eenheden in Groningen, Friesland en Drenthe*, Rijksuniversiteit Groningen, Groningen

Breaux, T. D. and Vail, M. W. (2006) Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. *Proceedings of the 14th IEEE International Requirements Engineering Conference* 46 - 55, IEEE Computer Society, Washington

Buchanan, W. J., Fan, L., Lawson, A., Schafer, B., Scott, R., Thuemmler, C., and Uthmani, O. (2010) *Interagency data exchange protocols as computational data protection law*, http://researchrepository.napier.ac.uk/id/eprint/3875, Edinburgh

Carabelea, C. and Boissier, O. (2005) *Autonomy in multi-agent systems*, Research Report 2005-700-004 edition, Saint-Etienne

Castelfranchi, C. (1995) Guarantees for autonomy in cognitive agent architecture. *Intelligent Agents Lecture Notes in Computer Science*, 56 - 70, Springer, Heidelberg

College bescherming persoonsgegevens (2004) *Informatieblad 23: Uw gegevens bij de politie*, Den Haag

Cozijn, C. (1996) *Wet en besluit politieregisters: een inventarisatie van knelpunten in de politiepraktijk*, Wetenschappelijk Onderzoek-en Documentatiecentrum, 's-Gravenhage

De Vey Mestdagh, C. N. J. (2008) Privacy en ICT. In De Vey Mestdagh, C. N. J., Dijkstra, J. J., and Huisjes, S. C. (eds.), *ICT-recht voor de praktijk*, 17 - 41, Wolters-Noordhoff, Groningen

De Vey Mestdagh, C. N. J. (1998) Legal Expert Systems. Experts or Expedients? The Representation of Legal Knowledge in an Expert System for Environmental Permit Law. *The Law in the Information Society, Conference Proceedings on CD-Rom* Firenze

De Vey Mestdagh, C. N. J. (2003) Administrative Normative Information Transaction Agents (ANITA): Legitimacy and Information Technology, the best of two worlds. *Access to knowledge and its enhancements, Proceedings ToKeN2000 symposium*, Delft University of Technology

De Vey Mestdagh, C. N. J. (1997) *Juridische kennissystemen rekentuig of rekenmeester?*, PhD Thesis, Rijksuniversiteit Groningen, Groningen

Dijkstra, P., Bex, F., and Prakken, H. (2005) Outline of a multi-agent system for regulated information exchange in crime investigations. *Argumentation in Artificial Intelligence and Law*, 27 - 38, Wolf Legal Publishers, Nijmegen

Dijkstra, P., Bex, F., Prakken, H., and De Vey Mestdagh, K. (2006) Towards a multi-agent system for regulated information exchange in crime investigations. *Artificial Intelligence and Law* 13:133 - 151.

Dijkstra, P., Prakken, H., and De Vey Mestdagh, K. (2007) An implementation of norm-based agent negotiation. *Proceedings of the 11th International Conference on Artificial intelligence and Law*, 167 - 175, ACM Press, New York

Doutre, S., McBurney, P., and Wooldridge, M. (2005) Law-Governed Linda as a semantics for agent interaction protocols (research abstract). *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005),* 1257 - 1258, Utrecht

Dung, P. M. (1995) On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77:321 - 357.

Euzenat, J. (2004) An API for ontology alignment. *Proceedings 3rd International Semantic Web Conference (ISWC)*, 698 - 712, Hiroshima

Feigenbaum, E. A. (1984) Knowledge engineering: the applied side of artificial intelligence. *Annals of the New York Academy of Sciences*, 426:91 - 107, New York

Gärdenfors, P. (1992) Belief revision: An introduction. *Belief revision* 29:1 - 28.

Gardner, A. L. (1987) *An artificial intelligence approach to legal reasoning*, MIT Press, Cambridge

Genesereth, M. R. and Ketchpel, S. P. (1994) Software agents. *Communications of the ACM* 37:7, 48 - 53.

Giblin, C., Liu, A. Y., Müller, S., Pfitzmann, B., and Zhou, X. (2005) Regulations expressed as logical models (REALM). In Moens, M. F. and Spyns, P. (eds.), *Proceeding of the 18th Annual Conference on Legal Knowledge and Information Systems* 37 - 48, IOS Press, Amsterdam

Gruber, T. R. (1993) A translation approach to portable ontology specifications. *Knowledge acquisition* 5:2, 199 - 220.

Hagerty, J. (2007) *SOX Spending for 2006*, AMR Research, Boston

Huang, J., Jennings, N. R., and Fox, J. (1995) Agent-based approach to health care management. *International Journal of Applied Artificial Intelligence* 9:4, 401 - 420.

Kielman, H. H. (2010) *Politiële gegevensverwerking en Privacy: Naar een effectieve waarborging*, PhD Thesis, Universiteit Leiden, Leiden

Koelewijn, W. (2009) *Privacy en politiegegevens: Over geautomatiseerde normatieve informatie-uitwisseling*, PhD Thesis, Universiteit Leiden, Leiden

Lessig, L. (1999) *Code and other laws of cyberspace*, Basic books, New York

McBurney, P. and Parsons, S. (2002) Games that agents play: a formal framework for dialogues between autonomous agents. *Journal of Logic, Language and Information* 11:3, 315 - 334.

McCarthy, J. (1980) Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence* 13:1-2, 27 - 39.

McCarty, L. T. (1980) The TAXMAN project: Towards a cognitive theory of legal argument. In Niblett, B. (eds.), *Computer science and law: an advanced course*, 23 - 43, Cambridge University Press, Cambridge

McNamara, P. and Prakken, H. (eds.) (1998) *Norms, Logics and Information systems: new studies in deontic logic and computer science*, IOS press, Amsterdam

Mommers, L., Koelewijn, W., and Kielman, H. (2007) Understanding the law: a method for legal knowledge dissemination, *Proceedings of the 11th International Conference on Artificial Intelligence and Law*. 195 - 203, ACM Press, New York

Parsons, S., Wooldridge, M., and Amgoud, L. (2002) An analysis of formal inter-agent dialogues. *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, 394 - 401, ACM Press, New York

Parsons, S., Wooldridge, M., and Amgoud, L. (2003) Properties and complexity of some formal inter-agent dialogues. *Journal of Logic and Computation* 13:347 - 376.

Perrussel, L., Doutre, S., Thévenin, J. M., and McBurney, P. (2007) A persuasion dialog for gaining access to information. In Rahwan, I., Parsons, S., and Reed, C. (eds.), *Argumentation in Multi-Agent Systems Lecture Notes in Computer Science*, 63 - 79, Springer, Hawaii

Prakken, H. (2006) Formal systems for persuasion dialogue. *The Knowledge Engineering Review* 21:02, 163 - 188.

Prakken, H. (2005) Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Computation* 15:6, 1009 - 1040.

Prakken, H. and Vreeswijk, G. (2002) Logics for defeasible argumentation. In Goebel, R. and Guenthner, F. (eds.), *Handbook of Philosophical Logic*, second edition, Vol 4, 219 - 318, Kluwer Academic Publishers, Dordrecht

Rahwan, I., Ramchurn, S. D., Jennings, N. R., McBurney, P., Parsons, S., and Sonenberg, L. (2004) Argumentation-based negotiation. *The Knowledge Engineering Review* 18:04, 343 - 375.

Rahwan, I. and Simari, G. (2009) *Argumentation in Artificial Intelligence*, Rahwan, I. and Simari, G. (eds.), Springer, Berlin

Rao, A. S. and Georgeff, M. P. (1995) BDI agents: From Theory to Practice. *Proceedings of the 1st International Conference on Multi-Agent Systems* 312 - 319, San Francisco

Reiter, R. A. (1980) Logic for default reasoning. *Artificial Intelligence* 13:81 - 132.

Richardson, S. and Asthana, S. (2006) Inter-agency Information Sharing in Health and Social Care Services: The Role of Professional Culture. *British Journal of Social Work* 36:4, 657 - 669.

Ruth, A. G. P. and Schreuders, E. (2000) *Politiegegevens beschermd, een toelichting op het gesloten verstrekkingenregime van de Wet politieregisters*, Registratiekamer, Den Haag

Sartor, G. (2010) Doing justice to rights and values: teleological reasoning and proportionality. *Artificial Intelligence and Law* 18:2, 175 - 215.

Schreuders, E. and Wel, H. (2005) *Evaluatie wet bijzondere politieregisters. Deel II. De Wet bijzondere politieregisters in de praktijk.*, WODC, Den Haag

Shoham, Y. (1993) Agent-oriented programming. *Artificial Intelligence* 60:1, 51 - 92.

Stranieri, A., Zeleznikow, J., Gawler, M., and Lewis, B. (1999) A hybrid rule - neural approach for the automation of legal reasoning in the discretionary domain of family law in Australia. *Artificial Intelligence and Law* 7:2-3, 153 - 183.

Subwerkgroep Criminele Inlichtingen Eenheden (2002) *Criminele Inlichtingen Eenheden - Resultaten van de Abrio-werkgroep*, Abrio, Woerden

Sycara, K. P. (1998) Multiagent systems. *AI magazine* 19:2, 79 - 92.

Teepe, W. G. (2007) *Reconciling Information Exchange and Confidentiality. A Formal Approach*, PhD Thesis, Rijksuniversiteit Groningen, Groningen

Torroni, P., Gavanelli, M., and Chesani, F. (2007) Argumentation in the Semantic Web. *IEEE Intelligent Systems* 22:6, 66 - 74.

Van der Velde, T. (2003) *RBS handleiding*, Regiopolitie Groningen, Groningen

Van Engers, T. M., Gerrits, R., Boekenoogen, M., Glassée, E., and Kordelaar, P. (2001) POWER: using UML/OCL for modeling legislation - an application report. *Proceedings of the Eighth International Conference on Artificial Intelligence and Law.* 157 - 167, ACM Press, New York

Van Veenen, J. and Prakken, H. (2006) A protocol for arguing about rejections in negotiation. In Parsons, S., Maudet, N., Moraitis, P., and Rahwan, I. (eds.), *Argumentation in Multi-Agent Systems Lecture Notes in AI 4049*, 138 - 153, Springer, Berlin

Vreeswijk, G. A. W. (2006) An algorithm to compute minimally grounded and admissible defence sets in argument systems. In Dunne, P. E. and Bench-Capon, T. J. M. (eds.), *Proceedings of the 2006 Conference on Computational Models of Argument* 109 - 120, IOS Press, Amsterdam

Walton, D. N. and Krabbe, E. C. W. (1995) *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*, State University of New York Press, Albany

Weiss, G. (1999) *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, Cambridge

Werkgroep CIE (2003) *Werkprocessen inwinnen van informatie door middel van informanten en leveren ZWACRI-informatie*, Abrio, Woerden

Wooldridge, M. (2002) *An Introduction to MultiAgent Systems*, John Wiley & Sons, Ltd, Chicester

Wooldridge, M. and Jennings, N. R. (1995) Intelligent agent: theory and practice. *Knowledge Engineering Review* 10:115 - 152.

Wooldridge, M. and Parsons, S. (2000) Languages for negotiation. *Proceedings of the Fourteenth European Conference on Artificial Intelligence* 393 - 400.

Summary

In many organizations data has to be exchanged, an activity which is often regulated by law. In most cases there is a central institution (for example, the mother company or the ministry of Justice) that is interested in both optimal and legitimate data exchange, because it has to give account of the effectiveness and lawfulness of its operations to the outside world (for example, the shareholders or the parliament). Besides the central institution there are regionally or functionally distributed local institutions with their own interests. The central institutions take the interests at the distributed level into account by formulating legal norms and central policies, which give room for fine tuning in local policies and individual decisions.

The regulation of data exchange serves several objectives. On the one hand, the privacy of persons must be protected. On the other hand, the legitimate goals of the exchanging organizations must be served. Typically, organizations must balance the goal to execute their tasks by exchanging data with the goal to protect their interests at the local level by not exchanging confidential data, while staying within the law.

The balancing act between goals is caused by two characteristics of the applicable laws and regulations. First, the regulations give discretionary power to organizations in which types of situations organizations can decide (not) to exchange data. In those cases where it is permitted to exchange data, an organization makes a decision which is expected to further its goals. Second, the regulations use *open-textured* concepts, which can have different local interpretations (for example 'the proper execution of the police task'). A concept is open textured if it is impossible to determine beforehand which situations can be classified as an instance of the concept.

In practice, data exchanging organizations can engage in dialogues and use their own interpretations of the regulations to further their goals. Therefore, during the interactions a definitive balance is sought between the goals of the organizations, which may give rise to interesting dialogues. For example, when an organization requests another organization to exchange data, the responding organization has to determine if its interests are served. The responding organization can negotiate by stating conditions under which it is willing to exchange data (for example that the data cannot be exchanged with other organizations). Furthermore, when a responding organization refuses a request, the requesting organization can try to persuade the other organization to exchange data. Ideally, these dialogues guarantee

that an optimal and legitimate balance is found in the exchange of data characterized by different goals. However, in practice this ideal is not always realized: the norms from regulations are not well known and local policies are more aimed at the protection of local data, which has as consequences that only a part of what can be exchanged legitimately is actually exchanged and that sometimes data is exchanged illegitimately.

The distributed nature of many organizations suggests that it may be worthwhile to use multi-agent technology. A *multi-agent system* is a collection of artificial agents, where each agent has some degree of control over his actions and interacts with other agents to fulfill his own and/or the system's goals. A multi-agent system arguably provides a natural means to model organizations, where the agents represent the organization's members. Furthermore, organizations typically interact with each other to try to achieve their goals, which is also typical for agents in multi-agent systems.

The Dutch police domain provides a worthwhile example of the problem of regulated data exchange, as it illustrates that not always an optimal and legitimate balance is found in the exchange of data. This thesis has investigated the idea of utilizing multi-agent technology to support regulated data exchange between Dutch police departments.

Chapter 2 presented a description and analysis of regulated data exchange in the Dutch police domain. The analysis showed that regional police departments need to exchange data, because departments often need data held by other departments in order to solve crime cases. Furthermore, each regional police department has a criminal investigations unit for gathering intelligence data on severe crime. The analysis also showed that the criminal investigation units have to balance the goal to execute their appointed police task with the goal to protect their interests at the local level (i.e., their investigations and informants), while staying within the law. In practice, criminal investigation units do not find an optimal and legitimate balance between their goals because they do not know all the applicable regulations and prefer their goal to protect their own their interests by refusing to exchange data. Chapter 2 also made a description and analysis of the legal framework for regulated data exchange in the Dutch police domain, which was rewritten into an intermediate representation and formalized in Appendix 1. The analysis illustrated that the legislator intended a free flow of data within the Dutch police organization and has created many rules which obligate the exchange of data. To enable criminal investigation units to protect their investigations and informants, the legislator allows grounds for refusal as a legal way to refuse the exchange of data. It was the legislator's objective that in specific cases, the relevant interests are weighed against each other. However, the analysis of the Dutch police organization revealed

that in practice the grounds for refusal are most of the time used as default instead of using them with serious deliberation in specific cases. Since the grounds of refusals are used as default, only a part of the permitted data exchanges occurs in the Dutch police domain.

Chapter 3 described how a multi-agent system can model regulated data exchange between organizations and argued why the Dutch police organization is a representative problem domain. Dialogue examples from chapter 2 about regulated data exchange were used to clarify the functional requirements for a multi-agent system. The requirements are that the agents must have knowledge about the domain, are able to reason with their knowledge and are able to have the typical interactions in the Dutch police domain. The examples also illustrated that agents engage in dialogues to negotiate about and to persuade each other whether data can be exchanged. The analysis showed that three dialogues types are necessary to model regulated data exchange: information-seeking, negotiation and persuasion dialogues. For the police domain, the relation between the dialogue types was interpreted as starting as a negotiation dialogue, which may shift to an embedded persuasion. If the negotiation terminates successfully, an information-seeking dialogue starts and its termination also terminates the overall interaction.

Chapter 4 specified the argumentation system to model the internal inferences of an agent and the dialogue system to model the interaction between agents. The internal reasoning was modeled as defeasible reasoning and formalized with the ASPIC argumentation system. ASPIC was chosen because it supports defeasible reasoning and since the ASPIC inference engine can be used in the proof-of-concept implementation. Deontic modalities and agent goals are not part of the ASPIC logic. However, they are expressed to be compatible with the ASPIC inference engine and to support the regulated data exchange between CIE officers.

Whereas a dialogue system's communication protocol specifies all the allowed utterances in a dialogue, a policy specifies the best choices from those utterances. In case of dialogues between CIE agents about regulated data exchange, the best utterance is one that is expected to further the goal to execute the appointed police task and the goal to protect local investigations and informants. Chapter 5 specified two types of dialogue policies, viz. for negotiation and for persuasion. The negotiation policies help to balance the agent's goals when the agent is permitted to exchange data but there is a violation of his interests. An agent could simply refuse to exchange data based on the violation of interests. However, with the negotiation policy the agent tries to find a condition under which there will be no violation of his interests. If he succeeds then also his goal to execute the police task by exchanging data is fulfilled. The persuasion policies help to persuade the other agent to exchange data in case of rejection of a request for data. Since in the current

police practice the grounds of refusals are used as default, helping to persuade can help to allow for more permitted data exchanges in the Dutch police domain. The dialogue policies were made applicable to other domains were data is exchanged by incorporating assumptions, such as cooperativeness and protection of own interests. Moreover, the policies were parameterized, which allows them to be fine tuned for the chosen application domain.

Chapter 6 presented the specification and implementation of the multi-agent system architecture for the Dutch police domain. The architecture, which meets the requirements specified in chapter 3, was developed using typical examples of dialogues about regulated data exchange. In order for the architecture to be generalizable to other domains, all terminology specific to the Dutch police domain was abstracted.

The research aim of this thesis has been to investigate how theories from multi-agent systems research, AI & Law research, and argumentation theory research can be further developed and applied in a realistic problem domain to provide the basis for automated support of organizations in promoting their goals in the context of regulated data exchange. Based on the research, a multi-agent system architecture has been specified and implemented to illustrate that it can provide a basis to support data exchange in the Dutch police domain. The support of regulated data exchange was defined as facilitating fewer unlawful and more lawful data exchanges. However, to determine whether the proposed multi-agent system can genuinely support automated regulated exchange of data in the Dutch police practice, more user experiments in realistic scenarios are necessary.

This thesis contributes to the fields of multi-agent systems, argumentation, and AI & Law in their combination and application in a serious problem domain. The multi-agent system architecture combines and adapts several elements from the literature: defeasible argumentation for the agents' internal reasoning and a dialogue system to model the agent dialogues. The research has by way of a realistic case study provided support for the naturalness and applicability of ideas from the literature. Furthermore, the present research has proposed a novel view on the nature of dialogues in the context of regulated data exchange, i.e., as negotiation with embedded persuasion. An important contribution to the research literature has been the specification of dialogue policies to assist the agents to balance their goals. To the best of my knowledge this is the first research which has specified realistic negotiation and persuasion policies in the context of regulated data exchange. Another research contribution has been the definition of realistic knowledge update policies for agents during dialogues about regulated data exchange. Research on this issue has also identified a new research issue in designing dialogue systems for persuasion, by identifying the problem that during a dialogue between agents,

previous dialogue utterances can sometimes no longer be maintained as a result of knowledge updates at later stages in a dialogue.

In this research, terminology specific to the Dutch police was abstracted and the dialogue policies were parameterized. Therefore, this research can be used in other domains where the exchange of data is regulated (for example, the exchange of patient data). This thesis has shown that it is possible to develop a realistic and implementable multi-agent system architecture that can provide the basis for automated regulated data exchange.

Samenvatting

Veel organisaties wisselen gegevens uit en dat wordt vaak gereguleerd door wetgeving. In de meeste gevallen is er een centrale institutie (bijvoorbeeld het moederbedrijf of het ministerie van justitie) die geïnteresseerd is in zowel een optimale als een rechtmatige uitwisseling van gegevens. De reden hiervoor is dat de centrale institutie zich moet verantwoorden voor de effectiviteit en de rechtmatigheid van haar activiteiten aan de buitenwereld (bijvoorbeeld de aandeelhouders of het parlement). Naast de centrale institutie zijn er regionaal of functioneel verdeelde instituties met hun eigen belangen. De centrale instituties houden rekening met de lokale belangen bij het formuleren van centraal beleid en wettelijke normen, die ruimte laten voor het precies afstemmen in lokaal beleid en individuele beslissingen.

De regulering van gegevensuitwisseling dient meerdere doelen. Aan de ene kant moet de privacy van de personen worden beschermd. Aan de andere kant moeten de rechtmatige doelen van de gegevensuitwisselende organisaties worden gediend. Meestal wegen organisaties het doel om hun taken uit te voeren (door het uitwisselen van gegevens) af met het doel om hun belangen op lokaal niveau te beschermen (door het *niet* uitwisselen van vertrouwelijke gegevens), terwijl er binnen de wet moet worden gebleven.

De afweging tussen doelen wordt veroorzaakt door twee kenmerken van de geldende wet- en regelgeving. Ten eerste geeft de regelgeving discretionaire bevoegdheden aan organisaties om in bepaalde situaties te beslissen om wel of niet gegevens uit te wisselen. In die gevallen waarin het is toegestaan om gegevens uit te wisselen, neemt een organisatie een beslissing waarvan verwacht wordt dat die het behalen van de doelen bevordert. Ten tweede gebruikt de regelgeving vage begrippen, die verschillende interpretaties kunnen hebben (bijvoorbeeld 'de goede uitvoering van de politietaak'). In de praktijk kunnen gegevensuitwisselende organisaties deelnemen aan dialogen en hun eigen interpretaties van de regelgeving gebruiken om het halen van hun doelen te bevorderen. Daarom wordt er tijdens de interacties tussen organisaties een definitieve balans gezocht tussen de verschillende organisatiedoelen, en dat kan aanleiding geven tot interessante dialogen. Bijvoorbeeld, wanneer een organisatie een andere organisatie vraagt om gegevens uit te wisselen, dan moet de reagerende organisatie bepalen of haar belangen worden gediend. De reagerende organisatie kan gaan onderhandelen door

voorwaarden te stellen onder welke zij bereid is om gegevens uit te wisselen. Bovendien kan de verzoekende organisatie de andere organisatie proberen te overtuigen om gegevens uit te wisselen, als een reagerende organisatie een verzoek tot gegevensuitwisseling heeft afgewezen. In het ideale geval garanderen deze dialogen dat er tijdens de gegevensuitwisseling een optimale en rechtmatige balans tussen de doelen wordt gevonden. In de praktijk wordt dit ideaal niet altijd gerealiseerd. De normen van de regelgeving zijn niet goed bekend en het lokale beleid is meer gericht op de bescherming van lokale gegevens. Dit heeft als gevolg dat slechts een deel van wat rechtmatig kan worden uitgewisseld daadwerkelijk wordt uitgewisseld en dat soms gegevens onrechtmatig worden uitgewisseld.

De gedistribueerde aard van veel organisaties suggereert dat het de moeite waard kan zijn om multi-agenttechnologie te gebruiken. Een multi-agentsysteem is een verzameling van artificiële *agents*[22], waarbij elke *agent* een zekere mate van controle heeft over zijn acties en interacteert met andere *agents* om zijn eigen en/of zijn systeemdoelen te vervullen. Een multi-agentsysteem lijkt een natuurlijke manier te zijn om organisaties te modelleren, waarbij *agents* de leden van organisaties representeren. Bovendien interacteren organisaties meestal met elkaar om hun doelen te verwezenlijken, wat ook typisch is voor *agents* in een multi-agentsysteem.

De Nederlandse politieorganisatie levert een waardevol voorbeeld van het probleem van gereguleerde gegevensuitwisseling, omdat het illustreert dat niet altijd een optimale en rechtmatige balans wordt gevonden bij het uitwisselen van gegevens. Dit proefschrift heeft onderzocht hoe agenttechnologie gebruikt kan worden om gereguleerde gegevensuitwisseling tussen de Nederlandse politiekorpsen te ondersteunen.

Hoofdstuk 2 presenteerde een beschrijving en analyse van gereguleerde gegevensuitwisseling in het Nederlandse politiedomein. De analyse toonde aan dat politieregio's gegevens moeten uitwisselen voor het oplossen van misdaadzaken, omdat regio's vaak behoefte hebben aan gegevens die aanwezig zijn in andere politieregio's. Verder heeft elke politieregio een criminele inlichtingen eenheid voor het verzamelen van gegevens over zware criminaliteit. De analyse toonde ook aan dat criminele inlichtingen eenheden het doel om hun politietaak uit te voeren in balans moeten brengen met het doel om belangen te beschermen op lokaal niveau (dat wil zeggen, hun eigen onderzoeken en informanten), terwijl er binnen de wet gebleven moet worden. In de praktijk blijkt dat criminele inlichtingen eenheden niet

[22] Om verwarring te voorkomen wordt het Engelse "*agent*" gebruikt om te verwijzen naar een intelligente (software) agent en wordt de Nederlandse term "agent" gebruikt om te verwijzen naar een politieagent.

een optimaal en rechtmatig evenwicht tussen hun doelen vinden, omdat ze niet op de hoogte zijn van de geldende regelgeving en de voorkeur hebben voor het doel om hun eigen hun belangen te beschermen. Hoofdstuk 2 gaf ook een beschrijving van het juridisch kader voor gereguleerde gegevensuitwisseling in het Nederlandse politiedomein, die werd herschreven in een tussenrepresentatie en geformaliseerd in appendix 1. Uit de analyse blijkt dat de wetgever een vrij verkeer van gegevens binnen delen van de Nederlandse politieorganisatie als doelstelling had en daarom veel regels heeft opgesteld die de uitwisseling van gegevens verplicht. Om criminele inlichtingen eenheden de mogelijkheid te geven om hun onderzoeken en informanten te beschermen, heeft de wetgever weigeringsgronden beschreven om op een rechtmatige manier de verstrekking van gegevens te weigeren. De wetgever had als doelstelling dat in specifieke gevallen de relevante belangen tegen elkaar worden afgewogen. Echter, uit de analyse van de Nederlandse politieorganisatie is gebleken dat in de praktijk de weigeringsgronden in de meeste gevallen als standaard worden gebruikt in plaats van ze te gebruiken na een zorgvuldige overweging. Omdat weigeringsgronden worden gebruikt als standaard, vindt in het Nederlandse politiedomein maar een deel van de toegestane gegevensuitwisseling plaats.

In hoofdstuk 3 werd beschreven hoe een multi-agentsysteem gereguleerde gegevensuitwisseling tussen organisaties kan modelleren en werd betoogd waarom de Nederlandse politieorganisatie een representatief probleemdomein is. Dialoogvoorbeelden uit hoofdstuk 2 over de gereguleerde gegevensuitwisseling werden gebruikt om de functionele eisen voor een multi-agentsysteem te verhelderen. De vereisten zijn dat de *agents* kennis over het domein moeten hebben, in staat moeten zijn om te redeneren met hun kennis en in staat moeten zijn om te interacteren op manieren die typisch bij de Nederlandse politie voorkomen. De voorbeelden illustreerden ook dat politieagenten met elkaar in dialoog gaan om te onderhandelen en om elkaar te overtuigen of er wel of geen gegevens kunnen worden uitgewisseld. De analyse toonde aan dat drie dialoogtypen nodig zijn om gereguleerde gegevensuitwisseling te modelleren: informatieuitwisseling, onderhandeling en multi-agentsysteem overtuigingsdialogen. Voor het politiedomein werd de relatie tussen de dialoogtypen geïnterpreteerd als een onderhandeling, die kan overgaan naar een ingebedde overtuigingsdialoog. Als de onderhandeling succesvol eindigt, wordt een informatieuitwisselingsdialoog gestart en wanneer deze vervolgens is beëindigd, is ook de gehele dialoog beëindigd.

Hoofdstuk 4 specificeerde het argumentatiesysteem om de interne redeneringen van een agent te modelleren en het dialoogsysteem om de interactie tussen agenten te modelleren. Het intern redeneren werd gemodelleerd als weerlegbare argumentatie en geformaliseerd met het *ASPIC*-systeem. *ASPIC* werd gekozen omdat het weerlegbare argumentatie ondersteunt en omdat het redeneermechanisme van

ASPIC gebruikt kan worden in de *proof-of-concept* implementatie. Deontische modaliteiten en agentdoelen maken geen deel uit van de *ASPIC* logica. Echter, deze worden uitgedrukt als predicaten en met reificatie van de uitspraken in hun *scope*. Het dialoogsysteem om de interacties tussen *agents* te modelleren werd aangepast om compatible te zijn met *ASPIC* en om gereguleerde gegevensuitwisseling tussen criminele inlichtingen eenheden te ondersteunen.

Waar een communicatieprotocol van een argumentatiesysteem alle toegestane uitingen in een dialoog bepaalt, specificeert een dialoogstrategie de beste keuze uit de toegestane uitingen. In het geval van dialogen tussen de *agents* over de gereguleerde gegevensuitwisseling, is de beste keuze de uiting die de doelen van de *agent* bevordert. Voor de criminele inlichtingen eenheden zijn de doelen de goede uitoefening van de politietaak en de bescherming van lokale onderzoeken en informanten. Hoofdstuk 5 specificeerde dialoogstrategieën) voor de onderhandelings- en overtuigingsdialogen. De onderhandelingstrategie helpt de doelen van een *agent* af te wegen. Bijvoorbeeld, een *agent* is toegestaan om gegevens uit te wisselen, maar daarbij kunnen zijn doelen worden geschonden. De *agent* kan dan simpelweg weigeren om gegevens op basis van de doelschending uit te wisselen. Echter, met de onderhandelingstrategie gaat de *agent* op zoek naar een voorwaarde waaronder zijn doelen niet worden geschonden. Als hij hierin slaagt dan kan hij de gegevens alsnog uitwisselen waardoor het doel om de politietaak uit te voeren wordt bevorderd. De overtuigingstrategie kan bij een weigering tot uitwisseling van gegevens helpen de andere *agent* te overtuigen om gegevens alsnog te verstrekken. Gegeven dat in de huidige politiepraktijk verzoeken tot uitwisseling van gevoelige gegevens meestal worden geweigerd, kan ondersteuning bij het overtuigen er voor zorgen dat er meer gegevens in het Nederlandse politiedomein kunnen worden uitgewisseld. De dialoogstrategieën werden toepasbaar gemaakt op andere domeinen waar gegevens worden uitgewisseld door het incorporeren van assumpties, zoals de coöperativiteit en de bescherming van eigen belangen. Bovendien werden de dialoogstrategieën geparametriseerd, waardoor ze kunnen worden afgestemd op het gekozen toepassingsdomein.

Hoofdstuk 6 presenteerde de specificatie en implementatie van het multi-agent systeem voor het Nederlandse politiedomein. De architectuur, die voldoet aan de functionele eisen beschreven in hoofdstuk 3, is ontwikkeld met behulp van typische voorbeelden van dialogen over gereguleerde gegevensuitwisseling. Om de architectuur generaliseerbaar te maken naar andere domeinen, is alle terminologie geabstraheerd die specifiek is voor de Nederlandse politie.

Het doel van dit proefschrift was om te onderzoeken hoe theorieën uit de onderzoeksgebieden van multi-agentsystemen, kunstmatige intelligentie & recht en argumentatie verder kunnen worden ontwikkeld en toegepast in een realistisch

probleemdomein. Daarbij wordt de basis gecreëerd voor geautomatiseerde ondersteuning van organisaties voor het verwezenlijken van hun doelen in het kader van gereguleerde gegevensuitwisseling. Op basis van het onderzoek is er een architectuur van een multi-agentsysteem gespecificeerd. Dit systeem is geïmplementeerd om te illustreren dat het een basis kan bieden voor de ondersteuning van gegevensuitwisseling in het Nederlandse politiedomein. De ondersteuning van de gereguleerde gegevensuitwisseling werd gedefinieerd als het bevorderen van minder onrechtmatige en meer rechtmatige gegevensuitwisselingen. Echter, om te bepalen of het voorgestelde multi-agentsysteem de gereguleerde gegevensuitwisseling in de Nederlandse politie in de praktijk kan ondersteunen, zijn vervolgexperimenten in realistische scenario's noodzakelijk.

Dit proefschrift draagt bij aan de gebieden van multi-agentsystemen, argumentatie en kunstmatige intelligentie & recht door hun combinatie en toepassing in een realistisch probleemdomein. De architectuur van het multi-agentsysteem combineert een aantal elementen uit de literatuur: weerlegbare argumentatie voor het interne redeneren van de agenten en een dialoogsysteem om de dialogen tussen *agents* te modelleren. Het onderzoek heeft door middel van een realistische casestudy de toepasbaarheid van ideeën uit de literatuur aangetoond. Bovendien heeft het onderzoek een nieuwe visie op de aard van de dialogen in het kader van gereguleerde gegevensuitwisseling voorgesteld, namelijk als onderhandeling met ingebedde overtuigingsdialoog. Een belangrijke bijdrage aan de literatuur is de specificatie van dialoogstrategieën om de agenten te helpen hun doelen te af te wegen. Naar mijn weten is dit het eerste onderzoek dat realistische dialoogstrategieën heeft gespecificeerd in het kader van de gereguleerde gegevensuitwisseling. Een andere onderzoeksbijdrage is de definitie van realistische kennis-updatestrategieën gedurende dialogen. Onderzoek naar dit onderwerp heeft een nieuw aandachtspunt geïdentificeerd voor het ontwerpen van dialoogsystemen voor overtuigingsdialogen. Het nieuwe onderzoeksonderwerp is dat gedurende een dialoog tussen agenten, eerdere dialooguitingen soms niet meer kunnen worden gehandhaafd als gevolg van kennis-updates in latere stadia van een dialoog.

In dit onderzoek is de terminologie die specifiek is voor de Nederlandse politie geabstraheerd en zijn de dialoogstrategieëën geparametriseerd. Daarom kan dit onderzoek ook gebruikt worden in andere domeinen waar gegevensuitwisseling wordt gereguleerd (bijvoorbeeld de uitwisseling van patiëntgegevens). Dit proefschrift heeft aangetoond dat het mogelijk is om een realistisch en implementeerbaar multi-agentsysteem te ontwikkelen dat de basis biedt om gereguleerde gegevensuitwisseling te automatiseren.

Dankwoord

Allereerst wil ik mijn promotor Henry Prakken bedanken. Henry, dankzij jouw feedback en ondersteuning, ook nadat ik niet meer bij de RUG werkte, is het proefschrift afgerond. Jouw bijdrage aan ons eerste gezamenlijke artikel is de basis geweest voor dit proefschrift. Kees de Vey Mestdagh, ik wil je bedanken voor je samenwerking in het ANITA project en jouw hulp als copromotor om de juridische kant van AI & Law beter leren te begrijpen.

Ook wil ik graag mijn beoordelingscommissie, Trevor Bench Capon, Jaap Hage en Jaap van den Herik bedanken. In het bijzonder wil ik Jaap van den Herik bedanken, die waardevol commentaar heeft gegeven waarmee ik het proefschrift verder heb kunnen verbeteren. Bart Verheij, hoewel je niet in de leescommissie zat, heb je toch naar mijn proefschrift willen kijken en nuttige feedback gegeven. Daarnaast wil ik het NWO bedanken voor de middelen om dit onderzoek uit te voeren en mijn collega's van de universiteiten van Groningen, Leiden, Utrecht en Maastricht voor de prettige samenwerking in het ANITA project.

Verder wil ik graag Tom van der Velde (privacy officer van de Groningse politie) bedanken die ik samen met Wouter Teepe heb geïnterviewd. Daarnaast wil ik ook Paul Elzinga (informatie architect bij de Nederlandse politie) bedanken die mij veel heeft uitgelegd over de Nederlandse politie praktijk. Ook wil ik mijn collega's bij het Centrum voor Recht en ICT en de afdeling juridische theorie bedanken voor de prettige werkomgeving.

Floris Bex wil ik bedanken omdat we privé en tijdens ons werk veel aan elkaar gehad hebben. Met hem heb ik zowel overdag als ‘s nachts een goede tijd gehad.

Wendy wil ik bedanken, dat ze me is blijven steunen terwijl ik een lange periode twee banen had, vaak verhuisde en dus weinig tijd over had.

Ten slotte wil ik mijn familie en mijn twee goede vrienden en paranimfen, Alle Hoeksma en Hans Schat, bedanken.

Pieter Dijkstra
Leidschendam, 24 maart 2012.